25
mi computer
CURSO PRACTICO DEL ORDENADOR PERSONAL,
EL MICRO Y EL MINIORDENADOR
ADDRESS MANAGER
FINANCE MANAGER
PURCHASE LEDGER 48K
STOCK CONTROL 48K
SINCLAIR DATABASE
YOUR SPECTRUM
SALES
UNITS
5000
4375
3750
3125
2500
1875
1250
625
0
JAN
FEB
MAR
APR
MAY
JUN
JUL
AUG
SEP
OCT
NOV
DEC
SONY
TRINITRON
Editorial Delta, S.A.
150 ptas.

MI COMPUTER CURSO PRACTICO
DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona, y comercializado en exclusiva por Distribuidora Olimpia, S.A., Barcelona

Volumen III - Fascículo 25

Director: José Mas Godayol
Director editorial: Gerardo Romero
Jefe de redacción: Pablo Parra
Coordinación editorial: Jaime Mardones
Francisco Martín
Asesor técnico: Jesús Nebra

Redactores y colaboradores: G. Jefferson, R. Ford, S. Tarditti, A. Cuevas, F. Blasco

Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Paseo de Gracia, 88, 5.°, Barcelona-8
Tels. (93) 215 10 32 / (93) 215 10 50 - Télex 97848 EDLTE

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador,* se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-85822-94-3 (tomo 3)
84-85822-82-X (obra completa)
Depósito Legal: B. 52/1984

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 048407
Impreso en España - Printed in Spain - Junio 1984

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, Madrid-34.
Distribuye para Argentina: Viscontea Distribuidora, S.C.A., La Rioja 1134/56, Buenos Aires.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio Blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Ferrenquín a Cruz de Candelaria, 178, Caracas, y todas sus sucursales en el interior del país.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (96 fascículos más las tapas, guardas y transferibles para la confección de los 8 volúmenes) son las siguientes:

a) Un pago único anticipado de 16 690 ptas. o bien 8 pagos trimestrales anticipados y consecutivos de 2 087 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 3371872 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Distribuidora Olimpia (Paseo de Gracia, 88, 5.°, Barcelona-8), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Distribuidora Olimpia, en la forma establecida en el apartado b).

Para cualquier aclaración, telefonear al (93) 215 75 21.

No se efectúan envíos contra reembolso.

Con este fascículo se incluyen las portadillas correspondientes al segundo volumen (fascículos 13-24)





Paul Chave

Para seguir avanzando

Finalizada la etapa introductoria, iniciamos ahora un auténtico curso avanzado, donde se estudiarán exhaustivamente temas ya tratados y se abordarán otros nuevos

A los profesionales de la informática (operadores, programadores y analistas de sistemas) les resulta bastante difícil mantenerse al día en los últimos adelantos en esta área, pero estas personas al menos pueden esperar el apoyo de sus directores y de los proveedores de los equipos con los cuales trabajan. En cambio, ¿adónde pueden acudir, en busca de una orientación imparcial, los entusiastas de la informática, que aprenden por sí mismos y por sus medios, dedicando a ello su tiempo libre?

¿Qué hacer primero? ¿Ir de tiendas y comprarse un ordenador económico? Hay tantos que resulta difícil decidirse por un modelo determinado sin contar con un consejo de confianza. Y, habiéndose decidido por alguno, ¿qué hacer después? Quizá la máquina elegida ofrezca más de un lenguaje de programación. ¿Cuál de ellos es el más indicado para las necesidades del usuario? ¿Qué paquetes de software ofrecen la mejor relación calidad-precio? ¿Cartuchos o programas en cinta? ¿Debe el usuario adquirir una unidad de disco? ¿O bastará con una unidad de cintas de cassette? ¿Necesita realmente una impresora ahora, o podría esperar hasta que el precio de las impresoras más sofisticadas baje un poco? Si tiene hijos, ¿cómo hacer frente a los constantes deseos de éstos por jugar a un juego detrás de otro, si la motivación principal de la inversión fue favorecer su "alfabetización informática" y ayudarlos en las tareas escolares?

El revolucionario avance de la tecnología del ordenador ha sido tan rápido y se ha introducido hasta tal punto en nuestro entorno cotidiano (el hogar, la oficina, la fábrica y el coche) que muchos de nosotros la hemos rodeado de una cierta aureola mágica.

Es un hecho que cada vez se están entrenando más y más personas para operar ordenadores y terminales de ordenador. Pero existe una importante

El desafío del futuro
En menos de una década, el ordenador se ha convertido en una parte esencial del tejido de nuestra sociedad, y cada año los cambios y los adelantos tecnológicos son más y más amplios y profundos. Mantenerse al corriente de la situación exige un esfuerzo considerable, pero para el recién iniciado en la materia constituye un auténtico desafío. Un curso de estudios a domicilio como el que le viene ofreciendo *Mi Computer*, bien planificado, puede ser una gran ayuda

LOGO
Aunque básicamente se usa como un lenguaje de aprendizaje para niños, el LOGO es también muy satisfactorio para el programador avanzado

BASIC
Continúa siendo el lenguaje más popular para la informática personal

PASCAL
Se le considera uno de los más profesionales entre los lenguajes populares

FORTH
Está en una categoría intermedia entre los lenguajes de alto nivel y el lenguaje máquina; es muy popular entre los programadores experimentados

Intérprete
Es un método para convertir el BASIC en código de lenguaje máquina línea por línea

Compilador
Produce una copia separada de un programa en código de lenguaje máquina

Lenguaje máquina
Aunque difícil de escribir, el código de lenguaje máquina puede ser muy gratificante, ya que no se pierde tiempo en interpretar

Ensamblador
El lenguaje máquina se suele escribir en forma de lenguaje ensamblador, que en lugar de números emplea etiquetas y símbolos alfanuméricos

CPU

Lenguajes a elegir

Falta poco para que estemos en situación de abordar varios de los nuevos lenguajes de programación que pueden servir de alternativas al BASIC. También analizaremos los programas y las configuraciones que se requieren para hacer que la CPU ejecute un programa. El usuario avanzado de ordenadores personales suele elegir su lenguaje de trabajo según la naturaleza del problema o la aplicación que deba afrontar

Kevin Jones

diferencia entre entrenamiento y aprendizaje. Entrenar significa ir conociendo un trabajo de forma mecánica. Aprender supone prepararse para dar un salto más allá de los límites del mero trabajo manual, hacia una amplia comprensión de cómo trabajan los sistemas, de sus potencialidades y de sus limitaciones.

Para muchas personas que trabajan en la industria electrónica o en escuelas y colegios, la respuesta parece ser un curso programado de informática, expuesto de tal modo que resulte comprensible para todos desde el principio. Los manuales de instrucciones individuales para máquinas específicas no pueden proporcionar un panorama equilibrado que permita relacionar un tipo de ordenador con otro. Tampoco reseñarán los escollos ocultos en la multiplicidad de máquinas existentes, ni le aconsejarán cabalmente sobre cómo sacar el máximo provecho de su adquisición. Después de todo, ¿qué clase de fabricante sería aquel que anunciara gratuitamente los productos de sus competidores?

Seguir un curso a domicilio estructurado de manera adecuada, apoyado posiblemente con una clase semanal en una escuela especializada en cursos de informática general y programación de ordenadores, podría ser una forma conveniente y económica de obtener una sólida educación en esta nueva ciencia.

A través de un curso de este tipo, usted no sólo aprendería a programar y a manejar su ordenador personal, sino que adquiriría una visión más amplia de cómo se utilizan los ordenadores en la vida cotidiana. Además de proporcionar una formación en programación y en análisis básico de sistemas, le

ofrecerá un panorama de todos los ordenadores que están en uso en el momento en vez de concentrarse en la máquina que emplea. Debería incluir el estudio de los periféricos y los accesorios que existen para todas ellas, con una explicación de sus principios operativos. Para situar al ordenador en su contexto, se deben examinar en profundidad los campos a los cuales se aplica actualmente y el software que posibilita esas aplicaciones. Por último, el curso debería incluir elementos de lógica formal, de sistemas numéricos y un poco de historia de la informática y de los ordenadores. En resumen, un curso de estudio a domicilio tendrá que cubrir todos los temas que se abarcan en un curso convencional de informática.

A partir de este número nos hemos propuesto estructurar el material para que le resulte un curso de estas características. Basándonos en los conocimientos de BASIC que ha adquirido usted hasta ahora, complementados con lo que ya se expuso sobre gráficos y síntesis de sonido por ordenador, nos hemos propuesto enriquecer sus experiencias en este lenguaje y presentarle los otros lenguajes de alto nivel con que cuentan los microordenadores (PASCAL, FORTH, LOGO y C, p. ej.) al tiempo que le proporcionaremos una buena base sobre programación en código de lenguaje máquina, la llave maestra que abre toda la potencia de un microprocesador.

El conocimiento del código de lenguaje máquina nos permitirá examinar las formas en que se definen los lenguajes de más alto nivel. Entonces, cuando hayamos estudiado cómo trabajan los compiladores y los intérpretes, podremos amalgamar estas dos ramas del conocimiento para empezar a definir nuestro propio lenguaje y escribir un compilador para él.

Esto no quiere decir que abandonemos el BASIC. Hemos de analizar los refinamientos de este lenguaje que nos permitan elaborar después proyectos generadores de software para aplicaciones útiles y de juegos de aventuras y juegos que se basen en pantalla.

Además de las funciones internas del ordenador, exploraremos los métodos para manipulación de archivos, tanto en cinta como en disco flexible, valiéndonos de la experiencia adquirida en cuanto a definir estructuras de datos y jerarquías dentro de la memoria interna del ordenador. De este modo podremos ampliar la capacidad incluso del más pequeño de los ordenadores personales, convirtiéndolo en un serio sistema de procesamiento de la información.

Teniendo siempre presente que no basta con estudiar un tema aislado, consideraremos en profundidad la amplia gama de paquetes de software que existe en la actualidad (hojas electrónicas, procesadores de textos, administradores de bases de datos y similares) con la idea tanto de comprender su funcionamiento y sus métodos como de aprender más acerca de las técnicas de programación profesionales, con el fin de incluir éstas en nuestra propia programación.

Se dedicará alguna atención a la electrónica básica, examinando la función y el diseño de los componentes individuales y de las formas en que éstos se combinan para construir ordenadores y sus periféricos. También volveremos a ocuparnos, ahora ya más detenidamente, de las máquinas en sí mismas: de los microordenadores populares, personales y de gestión, y de sus periféricos, examinando sus especificaciones y evaluando su impacto sobre la informática en general. No olvidaremos tampoco el aspecto humano de la industria del ordenador. Las empresas y las personas que diseñan el software y las que construyen las máquinas, e incluso los usuarios de ordenadores que hayan hecho alguna especial contribución en este campo, tendrán en el curso un lugar dedicado a ellos.

Si le interesa consolidar sus conocimientos acerca de los ordenadores con el fin de ampliar sus perspectivas en el campo laboral, entonces un curso de estudios a domicilio como el que le estamos ofreciendo puede ser el sustituto eficaz del primero-segundo nivel de estudios regulares de informática. Puesto que permite que el estudiante avance a su propio paso, es de igual valor tanto para el alumno rápido como para aquellos que quizá necesiten un poco más de tiempo para captar lo que, al fin y al cabo, es una disciplina compleja.

Por último, si lo que desea es simplemente estar mejor informado acerca de una tecnología que está llamada a cambiar la sociedad en el transcurso de su vida, entonces esta obra le ofrecerá desde ahora una guía exhaustiva. Además de los fundamentos del estudio de la informática, analizaremos el impacto que la nueva tecnología ejercerá sobre toda la sociedad. ¿De qué manera el advenimiento de los ordenadores, al irrumpir en nuestra vida cotidiana, modificará las relaciones entre las personas? ¿Qué cambios políticos tendrán lugar como consecuencia de una "explosión de información" que ha hecho posible el microprocesador de bajo costo? Es difícil dar respuestas razonables a estas preguntas. Los artículos de la prensa y los programas de televisión tienden a trivializarlas; muchas publicaciones especializadas en informática parecen otorgarles mayor complejidad que la que en realidad entrañan. Nosotros seguiremos el camino del medio: le suministraremos los datos necesarios para que elabore sus propias respuestas a estos interrogantes.



Un salto hacia adelante
Su aparición se anunció a la prensa internacional a comienzos de 1984 pero su salida al mercado se planificó para la primavera. El Quantum Leap de Sinclair supone el fin de la larga vinculación de la empresa con el microprocesador Z80. Equipado, en cambio, con una versión del 68000 de Motorola de 32 bits, posee 128 Kbytes de RAM (con más de 512 Kbytes disponibles) y dos microdrives QL incorporados. Sinclair abandona además su característico BASIC de entrada de tecla única

Ian McKinnell

La mejor oferta

Hasta hace poco tiempo, las unidades de disco flexible y las cintas flexibles no estaban al alcance de la mayoría de los usuarios de ordenadores personales, pero hoy la situación ha cambiado

Contacto peligroso

Recuerde no colocar los discos flexibles cerca de objetos imantados. Incluso algo aparentemente tan inocuo como el teléfono contiene electroimanes (se utilizan para hacer sonar el timbre) y hasta el altavoz de un equipo de alta fidelidad doméstico posee unos muy potentes

Los microordenadores son herramientas sumamente versátiles para manipular datos. Sin embargo, la manipulación de información es de poco valor si no se dispone de un medio para almacenarla cuando los datos no se necesitan de momento o cuando se apaga el ordenador. Lo que se puede conseguir de diversas maneras. Quien esté enterado de lo que la informática personal puede hacer se habrá percatado de las limitaciones del cartucho de ROM y de la cinta normal de cassette como métodos de almacenamiento permanente, y deseará conocer esos otros servicios más refinados que sólo los discos magnéticos ofrecen.

Pero antes de analizar los méritos de los discos vamos a considerar los otros sistemas.

El cartucho

Este método de almacenamiento es muy poco útil para el programador. La mayoría de los cartuchos contienen un tipo de PROM (*Programable Read Only Memory*: memoria programable de lectura



Ian McKinnell

La unidad de disco del BBC
Para que se puedan utilizar unidades de disco de esta clase con el BBC Modelo B, primero se debe instalar la ROM DOS (*Disk Operating System*: sistema operativo de disco) en la propia máquina. Por el contrario, las unidades del tipo "inteligentes" ya vienen equipadas con un chip DOS

solamente) que sólo proporciona un medio de dar entrada a los datos en el ordenador, por lo general juegos escritos en un lenguaje máquina extenso y complicado, o facilidades extras como aplicaciones al BASIC. No obstante, los cartuchos pueden contener unas EEPROM (*Electrically Erasable PROM*: PROM que puede borrarse eléctricamente) que se pueden leer y en las que se puede escribir de modo similar a la RAM interna, pero que no son "volátiles", en el sentido de que, cuando se quitan del ordenador o cuando éste se apaga, retienen la información. Igualmente, existen cartuchos para algunos ordenadores que contienen chips de RAM CMOS (*Complementary Metal Oxide Semiconductor*: semiconductor de óxido metálico complementario) de bajo poder, que retienen la información merced a una pila que contienen.

El inconveniente del almacenamiento de EEPROM y RAM CMOS es que son caros.

La cinta de cassette

Proporcionadas originalmente porque las unidades de disco eran muy caras, las cintas de cassette siguen siendo, con mucho, el medio de almacenamiento más popular. Por lo general, un reproductor de calidad media será suficiente, aunque en el caso de algunos fabricantes (en especial Commodore y Atari) sólo es posible usar las unidades que ellos han diseñado especialmente.

Los programas y los datos se almacenan en forma binaria como archivos secuenciales a través de la facilidad de grabación normal de la unidad de cassette, utilizando tonos diferentes para representar los 0 y los 1. Por lo general, una información identificada, como puede ser el nombre del archivo (y hasta la dirección de la memoria interna de donde se copia el archivo), se graba primero, seguida del archivo propiamente dicho, de a un bit por vez en bloques de un byte a los que posteriormente se les da formato en segmentos de 256 bytes. Muchos ordenadores incorporan una facilidad para verificación de errores en cada segmento, denominada *suma de control*.

Las órdenes típicas son SAVE, para grabar los archivos, y LOAD, para reproducirlos y recuperarlos. Algunos sistemas proporcionan órdenes adicionales para cassettes para diversas funciones especiales, incluyendo una facilidad para leer una cinta y preparar un catálogo de los nombres de archivos almacenados, y formatos de órdenes para almacenar y recuperar distintos tipos de datos.

El bajo costo y el formato, fácilmente comprensible, de las órdenes para el almacenamiento en cinta de cassette se ven neutralizados por una cantidad de inconvenientes serios:

1. En la mayoría de los casos el usuario opera los mandos de la unidad de cassette de forma manual para almacenar y recuperar, y esto exige una cuidadosa sincronización de la pulsación de los botones y una determinación precisa del volumen.
2. Dado que la información se almacena secuencialmente, la recuperación de un archivo específico (excepto en el caso de la grabadora de cassette Hobbit, controlada por software, y del microcassette incorporado del Epson HX-20) implica o bien la cuidadosa supervisión de un contador de cinta de precisión (¡si es que se dispone de alguno!) para permitir el avance rápido-rebobinado hasta un punto justo antes del archivo deseado, o bien una búsqueda a cargo del ordenador del nombre del ar-

chivo empezando por el comienzo de la cinta. El almacenamiento secuencial también significa que es imposible almacenar eficazmente datos que requieran leerse por secciones pequeñas desde cualquier punto de un archivo sin procesar todo el archivo completo. El tipo de almacenamiento que puede conseguir esto se conoce como *de acceso directo* y es necesario para todo sistema eficaz de archivo de base de datos, como listados de direcciones o entradas de control de existencias.

3. Todo lo anterior, junto con la pequeña cantidad de bits que se almacenan-recuperan por segundo utilizando el almacenamiento en cassette (normalmente, entre 300 y 1 200 bits), supone que un sistema de cinta de cassette sea muy lento.

4. Aun cuando los datos se hayan grabado por primera vez en la forma correcta, éstos se pueden borrar tras haberlos reproducido una cantidad no determinable de veces, debido al desgaste que supone la cabeza de cinta.

5. Las características de las grabadoras varían de un fabricante a otro, y cabe que los datos grabados en un modelo no se puedan reproducir en otro.

El disco flexible

En comparación con los sistemas de almacenamiento en cassette y en cartucho, el almacenamiento en disco ofrece pocos inconvenientes importantes. Las unidades de disco flexible son de construcción compleja y delicada, y bastante caras. Los mismos discos flexibles son, asimismo, costosos. Pero a cambio el usuario consigue un método fiable, cómodo y rápido para almacenar grandes cantidades de datos, operando a una velocidad entre 50 y 200 veces mayor que la del almacenamiento y recuperación en cinta.

Todas las unidades de disco poseen un tipo de sistema operativo de disco (DOS: *Disk Operating System*), con una rutina que formatea la distribución de la información de un disco en pistas. Conviene resumir aquí cuanto fue expuesto en otro lugar de la obra (véanse pp. 324 y 325). Por lo general un disco dispone de 35 a 80 pistas por cada cara y cada pista está dividida en un número variable de arcos llamados sectores. En las pistas más cortas, próximas al centro del disco, hay menos sectores que en las exteriores. Cada sector consta de un bloque de datos, por lo general de 256 bytes.

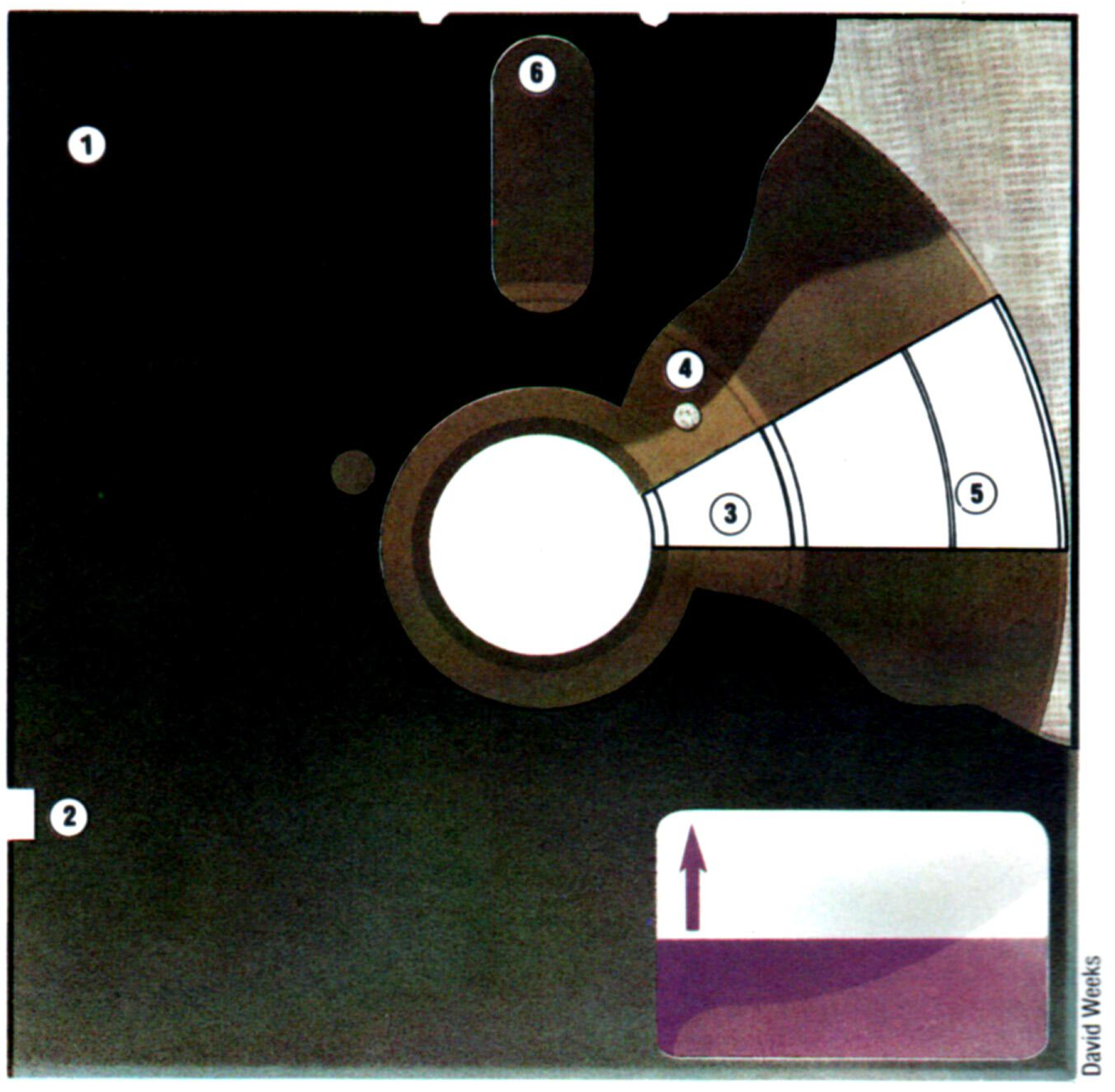
El DOS "recuerda" dónde está almacenada la información que contiene el disco. Esto normalmente se consigue mediante la creación de un mapa de disponibilidad de bloques (BAM: *Block Availability Map*), ya sea almacenado en el disco o retenido en la memoria, y un catálogo o directorio. El BAM contiene un registro de los bloques que ya están en uso y de los que están libres para un nuevo almacenamiento. El catálogo es una lista de los nombres de los archivos, tipos de archivo y las posiciones de pista y sector en que se encuentran. Por lo general se coloca en la pista central y se puede cargar en la memoria del ordenador para su consulta. El DOS posiciona la cabeza de lectura-escritura después de consultar el BAM, y cataloga y organiza el almacenamiento y la recuperación de los datos.

La disposición de la información en pistas y sectores y el posicionamiento exacto de la cabeza de lectura-escritura permite que el DOS ofrezca un tratamiento de acceso directo. Los datos se pueden grabar y extraer en fragmentos tan pequeños como de un byte por vez, si así se requiriera. Grosso modo, las diferencias entre las unidades de disco se reducen a la cantidad de datos que se pueden almacenar (que suelen oscilar entre los 100 y los 400 Kbytes), la velocidad a la cual se pueden transferir los datos, y los medios a través de los cuales el usuario puede controlar el almacenamiento y la recuperación utilizando el DOS.

Existen tres métodos principales para implementar un DOS. El más eficaz consiste en incluirlo en forma de ROM dentro de la unidad de disco, bajo el control del propio microprocesador de la unidad con la RAM asociada. Se dice que esta unidad de disco es una unidad "inteligente": al recibir una instrucción proveniente del procesador central, puede procesar independientemente complejas rutinas para manipulación de disco, permitiendo que el procesador continúe con la ejecución de un programa. Todas las actuales unidades de disco Commodore son inteligentes en este sentido y al operar no utilizan nada de la memoria interna del ordenador.

Un sistema más popular es el tipo que carga el DOS desde el disco a la RAM del ordenador mediante una orden o de forma automática, cuando se enciende el ordenador. El tercer método incluye una forma de DOS en el propio sistema operativo del ordenador. Los Spectrum poseen esta facilidad y la Acorn Computers suministra un DOS para el BBC Micro denominado *Disk Filling System* (sistema de llenado de discos) que proporciona un control limitado sobre el disco. Las rutinas para manipulación de discos incluyen órdenes SAVE y LOAD, una orden CAT (o directorio), una orden para formatear el disco (o cartucho de cinta) y varias órdenes de acceso directo y creación, manipulación y borrado de archivos secuenciales.

El diskette
Los diskettes se componen de Mylar, o una lámina plástica similar, elástica y resistente al desgaste, revestida de un óxido metálico capaz de retener una carga magnética. Encerrado dentro de un contenedor protector cuadrado y de plástico, el disco gira sobre el eje. La cabeza de lectura-escritura accede a la superficie de grabación a través de la ranura que se ve en la parte inferior de la ilustración



1 CONTENEDOR PROTECTOR
2 RANURA DE PROTECCIÓN
3 SECTOR
4 AGUJERO DE ALINEACIÓN
5 PISTA
6 RANURA DE ACCESO

Atacado por hormigas

La importancia de "Ant attack" no reside sólo en la notable calidad de sus gráficos, sino en la sutil aplicación del algoritmo que genera el escenario de la acción, similar a un laberinto

Los escritores y los editores de software jamás han estado de acuerdo con la protección que les brindan las leyes de propiedad intelectual y esta insatisfacción se ha manifestado en los diversos intentos que han realizado con el fin de impedir la copia de sus programas. El autor de este juego, Sandy White, ha intentado, mediante la utilización de otro recurso, evitar que se plagie su obra: ha solicitado patente de privilegio de la técnica de software que produce los gráficos en pantalla. Dado que la Ley de Patentes británica de 1977 niega específicamente la protección de esta clase de programas para ordenador (aduciendo que no se pueden considerar como inventos), se concluye que la patente en cuestión cubre una fórmula matemática o un algoritmo.

Este hecho es interesante, porque no suelen necesitarse algoritmos complejos en un juego de este tipo. ¿Y qué tiene *Ant attack* (Ataque de las hormigas) que exija la creación de nuevas normas encaminadas a la protección de la propiedad intelectual del software?

Ant attack también es inusual en el sentido de que no desciende directamente de ningún juego recreativo. La mayoría de los juegos populares para ordenadores personales tienen sus raíces en las concepciones de Atari, Taito y los otros fabricantes de máquinas exclusivas para juegos. A *Ant attack* lo concibió un graduado del Edinburgh College of Art, que ha manifestado su desconocimiento de la tradición de juegos recreativos. Sandy White nunca antes había escrito software para juegos y todos sus esfuerzos por documentarse en este campo se limitaron a preguntarles a sus amigos qué era lo que les gustaba de esa clase de juegos.

Sorprendentemente, su paquete, notablemente vanguardista, fue rechazado por Sinclair Research, quienes no pudieron evaluar la videocinta de *Ant attack* que White les enviara porque, según le dijeron, ¡no tenían ningún aparato de videocassette!

1 ¡Mi héroe!
En la primera pasada por el juego, la "víctima" está convenientemente situada junto a las puertas de acceso a la ciudad. Un rápido salto por encima de la muralla protectora y el protagonista (hombre o mujer) es recibido con el grito de "¡Mi héroe, llévame lejos de todo esto!"

2 Situación comprometida
En algunas ocasiones, el hecho de que las hormigas no puedan trepar escaleras resulta de gran utilidad (aunque la razón por la cual nuestro héroe haya subido tan alto aún está por descubrirse). Trepar por obstáculos como éste le permite al protagonista lanzar granadas contra las hormigas guerreras sin temor de que las devuelvan; pero recuerde que está jugando contra reloj

La primera característica novedosa de este juego de laberinto con que se encontrará el usuario es que permite que el jugador escoja el sexo del protagonista principal. Y pisándole los talones viene el primer descuido. Sin tener en consideración el sexo elegido por el usuario, el fotograma que abre el juego, que lo sitúa en el escenario en unas 30 palabras o así, le explica cómo ha llegado a sus oídos una llamada de socorro "irresistible para un héroe como usted".

El protagonista, acosado por monstruosas hormigas, se puede defender solo (o, por supuesto, sola) arrojando granadas. Lamentablemente, no hay coherencia en cuanto al efecto que producen estas granadas en las hormigas. Si bien esto podría ser resultado de un factor de azar deliberado, es más probable que sea consecuencia de una programación indiscriminada. Desplazar al protagonista 90 grados en sentido contrario a las agujas del reloj se consigue pulsando la tecla M del Spectrum y la tecla contigua, SYMBOL SHIFT, hace girar la figura en sentido contrario. Las teclas tipo membrana de plástico moldeado del Spectrum no proporcionan control adecuado sobre esta transformación, que invariablemente produce frustración en el jugador.



Cortesía de Soft

Ian McKinnell

Al parecer, *Ant attack* se desarrolló antes de que se lanzara al mercado la interface 2 de Sinclair, que acepta dos palancas de mando estándar de Atari. El juego mejoraría muchísimo si se lo actualizara para utilizar estos periféricos, si bien necesitaría dos palancas de mando para manipular la estructura de órdenes.

Además de hacer girar el distintivo, desplazarlo hacia adelante, hacerlo saltar o arrojar granadas (se puede escoger entre cuatro distancias de lanzamiento), el jugador puede elegir uno de cuatro puntos de vista, cada uno de ellos centrado en el distintivo.

Es esta sección de la generación de gráficos del programa lo que lo diferencia de la mayoría de los otros juegos que ocupan menos de 48 Kbytes. La transformación es virtualmente instantánea, eclipsando por completo la ejecución normal de los generadores de gráficos tridimensionales que existen para el Spectrum. La capacidad de cambiar los puntos de vista es esencial para el juego. Sin ella, una considerable porción del campo de juego quedaría a menudo oculta a la vista.

Comprensiblemente, el autor se muestra reacio a revelar demasiadas cosas acerca de los métodos de trabajo que han adoptado él y su colaboradora Angela Sutherland. No obstante, ha dado a entender que el campo de juego no está retenido, como uno esperaría, como una matriz de 128 × 128 × 6. La prueba de ello es evidente si, en vez de entrar en la ciudad, se hace que el distintivo del jugador gire en redondo y se encamine hacia el desierto. Después de una breve caminata, él o ella llegan a otra ciudad, y luego a otra, y así sucesivamente.

Y así llegamos al objetivo del juego en sí mismo. Éste está ambientado en la ciudad de Antescher (así llamada como homenaje de los autores del juego al artista y diseñador holandés M. C. Escher, quien dibujó ingeniosas estructuras ilusorias que en realidad eran imposibles de construir). Parado frente a sus puertas, el jugador escucha los gritos de una persona en peligro, salta por encima de la muralla baja y cae dentro de la ciudad, donde emprende la búsqueda de la víctima, saltando obstáculos o girando para evitarlos y seguir adelante. La ciudad aparece en proyección isométrica y no se realiza ningún intento por ser fiel a la perspectiva.

Sólo una pequeña porción de la ciudad está a la vista en cada momento y los fotogramas se mueven a lo ancho a medida que la figura se desplaza hacia la izquierda, la derecha, arriba o abajo. El *scrolling* es excelente, al igual que la animación de las figuras. Es también notable el sentido del humor de que se hace gala en la animación.

Enseguida se hace evidente que la ciudad está habitada por inmensas hormigas cuya mordedura, aunque no inmediatamente fatal, provoca la muerte si se recibe una cantidad suficiente. Si una hormiga se percata de la presencia del jugador se dispone a seguirlo. Con cierta dosis de habilidad se la podrá quitar de encima, de lo contrario tendrá que recurrir a la nada fiable granada. Y, cuidado, no vaya a lanzarla contra la pared que tiene inmediatamente delante, porque podría volar usted por los aires.

En la primera pasada a través del juego, la figura a rescatar está a la vista, al otro lado de la entrada. En las sucesivas pasadas se va haciendo más difícil de hallar y más difícil de alcanzar. Invariablemente está situada sobre el nivel del suelo. El rescatador puede saltar sólo un nivel cada vez, de modo que si la víctima no está directamente accesible desde el suelo (mediante una escalera, p. ej.), el rescatador tiene un auténtico problema. La única forma es esperar a que las hormigas ataquen en un punto apropiado, paralizar una y saltar sobre el insecto, utilizando su cuerpo como primer peldaño hacia arriba.

El rescatador también puede, de esta forma, "poner una pierna" sobre la víctima, en caso de que fuera necesario (las hormigas nunca atacarán a la víctima). La pasada termina cuando tanto el rescatador como la víctima están fuera de la ciudad.



Ian McKinnell

Los creadores del juego
Ant attack fue el primer intento de su autor, Sandy White, por escribir software comercial. Sandy, que sólo tenía 23 años cuando el paquete salió al mercado por primera vez, a fines de 1983, acababa de graduarse en el Edinburgh College of Art con una titulación en escultura cuando concibió la idea de crear un programa de juegos para microordenadores personales. Una amiga suya, Angela Sutherland, colaboró en el diseño de las estructuras que componen la ciudad de Antescher

El enigma de los arenales

Este plano general de la ciudad de Antescher se levantó después de fotografiar concienzudamente la pantalla del monitor cientos de veces, haciendo un complicado fotomontaje de los resultados y solicitándole luego a un artista su impresión acerca del escenario. Los autores White y Sutherland les han puesto nombre a las principales estructuras, realizando juegos de palabras casi intraducibles con el término *ant* (hormiga). Observe también cómo en la esquina izquierda ¡aparece el copyright: © SW!

1. THE WATCHTOWER
2. PHOSPHOR HENGE
3. THE QUAI
4. TABLE ANTCHAIR
5. THE FORUM
6. THE ANTICHAMBER
7. SKAZ YANDOR
8. THE PYRAMID
9. THE ANCIENT
10. OXYMINE
11. THE MONUMENT
12. ANT EDEN
13. ARGON'S LEAP
14. ARTANT'S VILLA
15. THE ANTIMATTER CUBE
16. DROXTRAP
17. ADRIANT'S WALL
18. BONZAI WALK
19. THE SQUARENA
20. THE CRYPT

Cortesía de Your Spectrum

Kevin Jones

Álgebra para la toma de decisiones

Los ordenadores efectúan sus funciones haciendo pasar una serie de voltajes altos o bajos por los circuitos electrónicos. Estos voltajes pueden representarse por los bits 1 y 0

El álgebra de Boole, rama de las matemáticas que aplica la lógica verdadero-falso, es la base teórica a partir de la cual se realiza físicamente la arquitectura del ordenador. Los conceptos y las leyes del álgebra booleana son pocos y fáciles de comprender.

En estas páginas estudiaremos con detalle los aspectos teóricos y prácticos del diseño de circuitos lógicos, con ejemplos de circuitos básicos que son los que trabajan dentro de un ordenador personal. Las leyes del álgebra booleana se basan en tres operaciones lógicas simples: AND, OR y NOT (véanse pp. 68 y 69). Estas tres operaciones lógicas siguen muy de cerca la forma en que se emplean dichas tres partículas en el lenguaje cotidiano. Observemos esta oración:

> Si hace buen tiempo AND(y) es sábado, David saldrá a dar un paseo.

Que David salga o no a dar un paseo depende de dos cosas: que haga buen tiempo y que sea sábado. Para tomar una decisión respecto a si salir o no de paseo, David considerará si los enunciados "hace buen tiempo" y "es sábado" son verdaderos o falsos. Cuatro son las posibles combinaciones y sólo una de ellas hará que David salga a dar un paseo. Una tabla que muestre todas las combinaciones posibles de una serie de enunciados es una *tabla de verdad*. He aquí la tabla de verdad para nuestra AND lógica:

Hace buen tiempo	Es sábado	David saldrá a dar un paseo
FALSO	FALSO	FALSO
FALSO	VERDADERO	FALSO
VERDADERO	FALSO	FALSO
VERDADERO	VERDADERO	VERDADERO

Un proceso similar ilustrará la función de la operación lógica OR. Consideremos esta oración:

> Si José OR(o) Ana van al partido, Juan irá también.

Nuevamente hay dos condiciones que determinarán que Juan vaya o no al partido: que José pueda ir, *o* que Ana pueda ir. Al igual que con AND, podemos construir una tabla de verdad para el nexo OR. Dado que hay dos condiciones, cada una de las cuales puede ser verdadera o falsa, otra vez son cuatro las combinaciones posibles. La tabla de verdad para la enunciación será la siguiente:

José irá	Ana irá	Juan irá al partido
FALSO	FALSO	FALSO
FALSO	VERDADERO	VERDADERO
VERDADERO	FALSO	VERDADERO
VERDADERO	VERDADERO	VERDADERO

La tercera operación lógica (NOT) realiza una función muy sencilla. Consideremos esta proposición:

> Si NOT(no) está oscuro, saldré.

Esta vez la única condición a considerar es si está oscuro. Esto puede ser verdadero o falso y, por tanto, para nuestra tabla de verdad sólo hay dos condiciones posibles:

Está oscuro	Saldré
FALSO	VERDADERO
VERDADERO	FALSO

Puertas lógicas

Los dispositivos electrónicos que componen los circuitos lógicos de un ordenador se denominan *puertas lógicas*. Las tres puertas lógicas más simples imitan la función de las operaciones lógicas AND, OR y NOT. Estas puertas funcionan representando una condición VERDADERO mediante el dígito binario 1, y la condición FALSO mediante el dígito binario 0. Así, para cada puerta lógica podemos construir una tabla de verdad con todas las combinaciones de entradas junto con la salida resultante.

La tabla de verdad y el diagrama para la puerta ADN con entradas A, B y salida C son:

A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

La función de la puerta AND se puede describir en palabras como: "la salida será 1 si ambas entradas son 1; si no, será 0". La notación booleana para la salida de una puerta AND es A.B. La tabla de verdad y el diagrama para la puerta OR son:

A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

Liz Dixon



La puerta OR se puede resumir con la siguiente frase: "la salida será 1 si alguna de las entradas, o ambas, es 1". La expresión booleana para la salida de una puerta OR es A+B.

A diferencia de AND y OR, la puerta NOT sólo posee una entrada y una salida. Su tabla de verdad es la más simple de las tres:

A	B
0	1
1	0

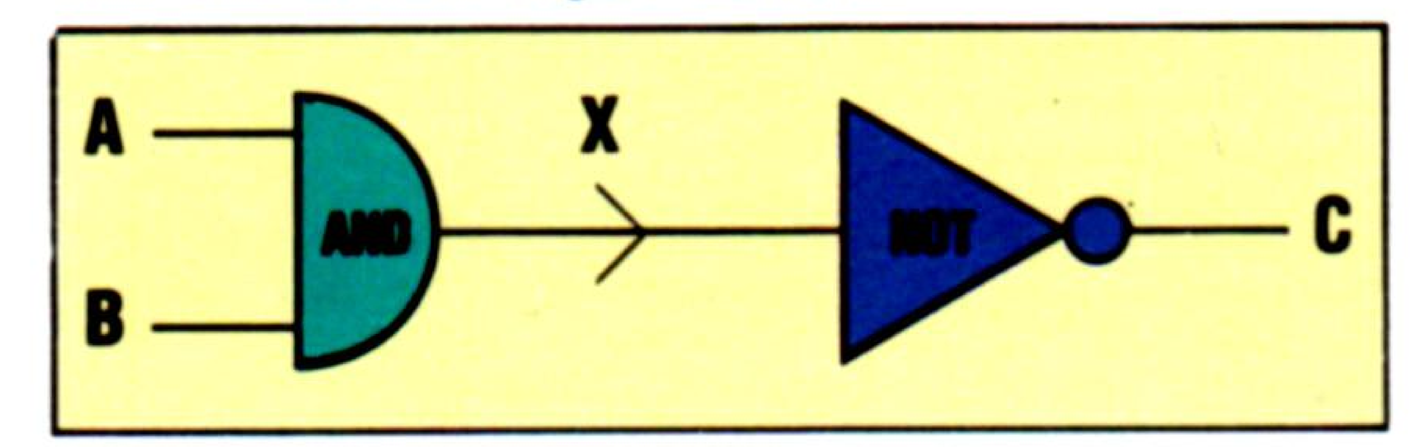
LA PUERTA NOT

A —NOT— B

La puerta NOT se resume así: "la salida será lo contrario de la entrada". La expresión booleana para la salida de una puerta NOT es $\bar{A}$.

Combinando puertas lógicas

Podemos unir entre sí varias puertas lógicas para obtener circuitos lógicos secuenciales y combinados. Éstos, a su vez, se combinan para producir la arquitectura del ordenador. Todo circuito lógico se puede representar mediante una tabla de verdad que describa qué salida se puede esperar para cualquier posible combinación de entradas. Observemos este circuito lógico sencillo:

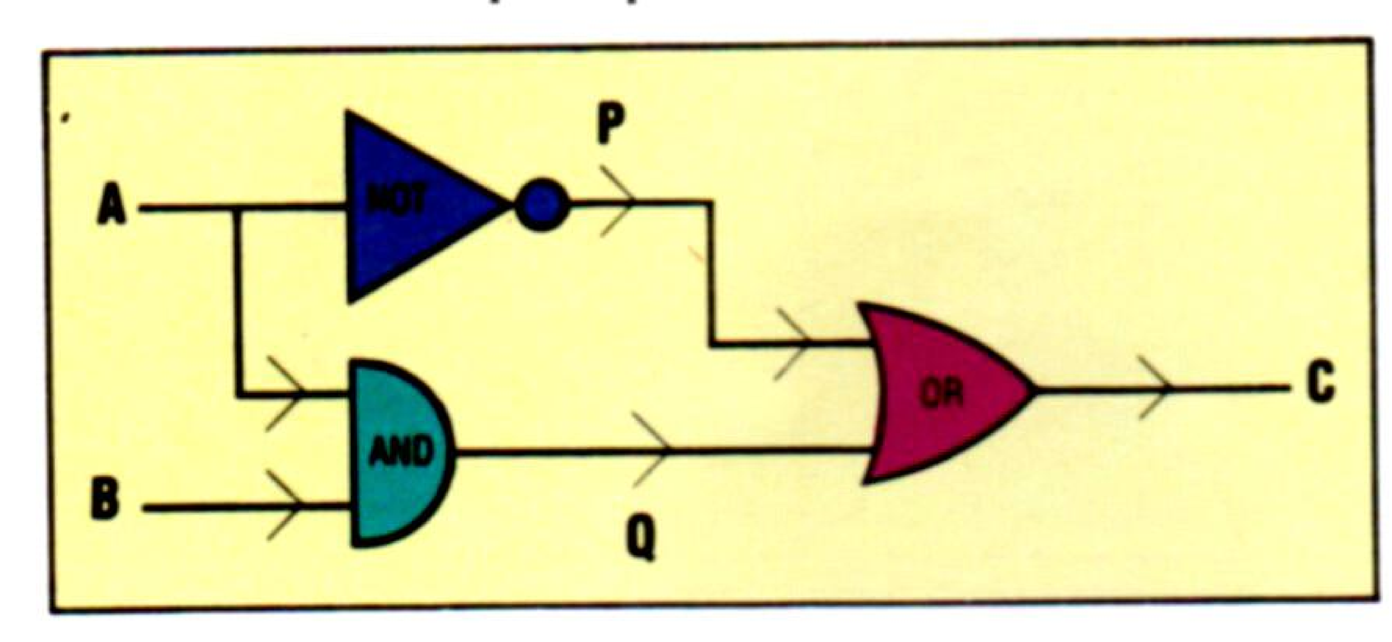


En este circuito hay dos entradas, A y B, y una salida, C. Para ayudar a construir la tabla de verdad para el circuito, hemos llamado X a la salida de la primera puerta. Como hay *dos* entradas para el circuito, ello significa que hay cuatro posibles combinaciones de entradas (o sea, 2^2).

A	B	X	C
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

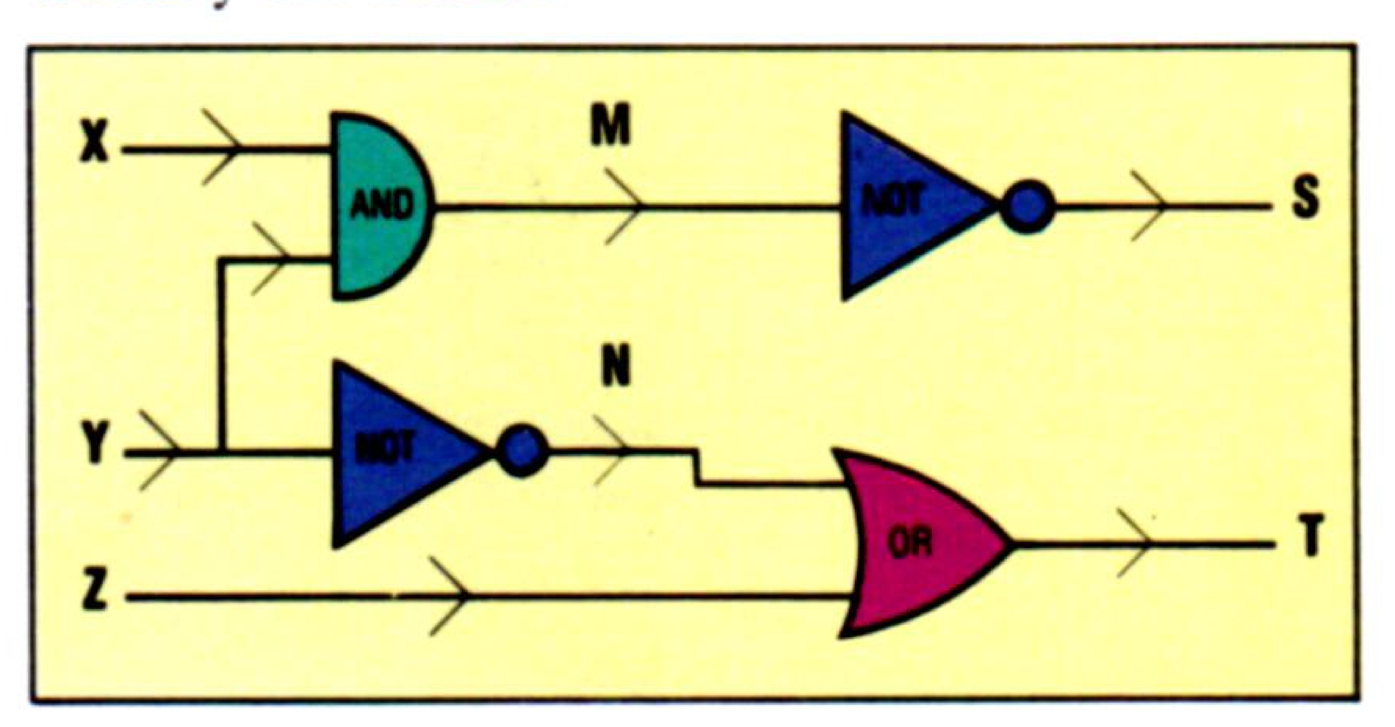
La salida de la puerta AND, X, se hace pasar por la puerta NOT para producir la salida final, C.

He aquí un circuito más complicado y su tabla de verdad. Como sólo hay dos entradas, las combinaciones de entradas posibles siguen siendo cuatro (2^2). La segunda mitad de esta tabla de verdad (las columnas P, Q y C) es un acomodo de parte de una tabla de verdad para puerta OR.



A	B	P	Q	C
0	0	1	0	1
0	1	1	0	1
1	0	0	0	0
1	1	0	1	1

La utilización de tablas de verdad no se limita a circuitos de dos entradas y una salida, sino que se pueden ampliar para cualquier circuito. Veamos a continuación un ejemplo de un circuito de tres entradas y dos salidas.



Como en este circuito hay *tres* entradas, debemos considerar ocho posibles combinaciones (o sea, 2^3):

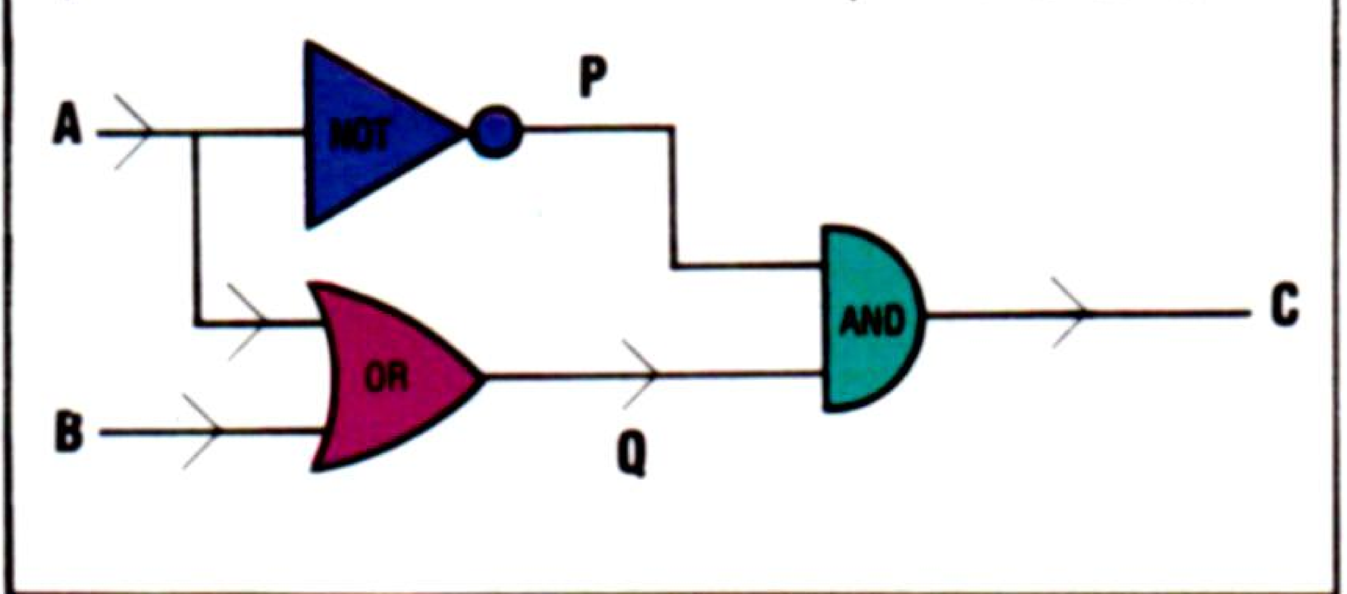
X	Y	Z	M	N	S	T
0	0	0	0	1	1	1
0	0	1	0	1	1	1
0	1	0	0	0	1	0
0	1	1	0	0	1	1
1	0	0	0	1	1	1
1	0	1	0	1	1	1
1	1	0	1	0	0	0
1	1	1	1	0	0	1

EJERCICIO 1

1) Construir una tabla de verdad para la siguiente situación: "María podría conducir un coche si hubiera aprobado su examen de conducir 0 (OR) si fuera acompañada por un conductor cualificado".

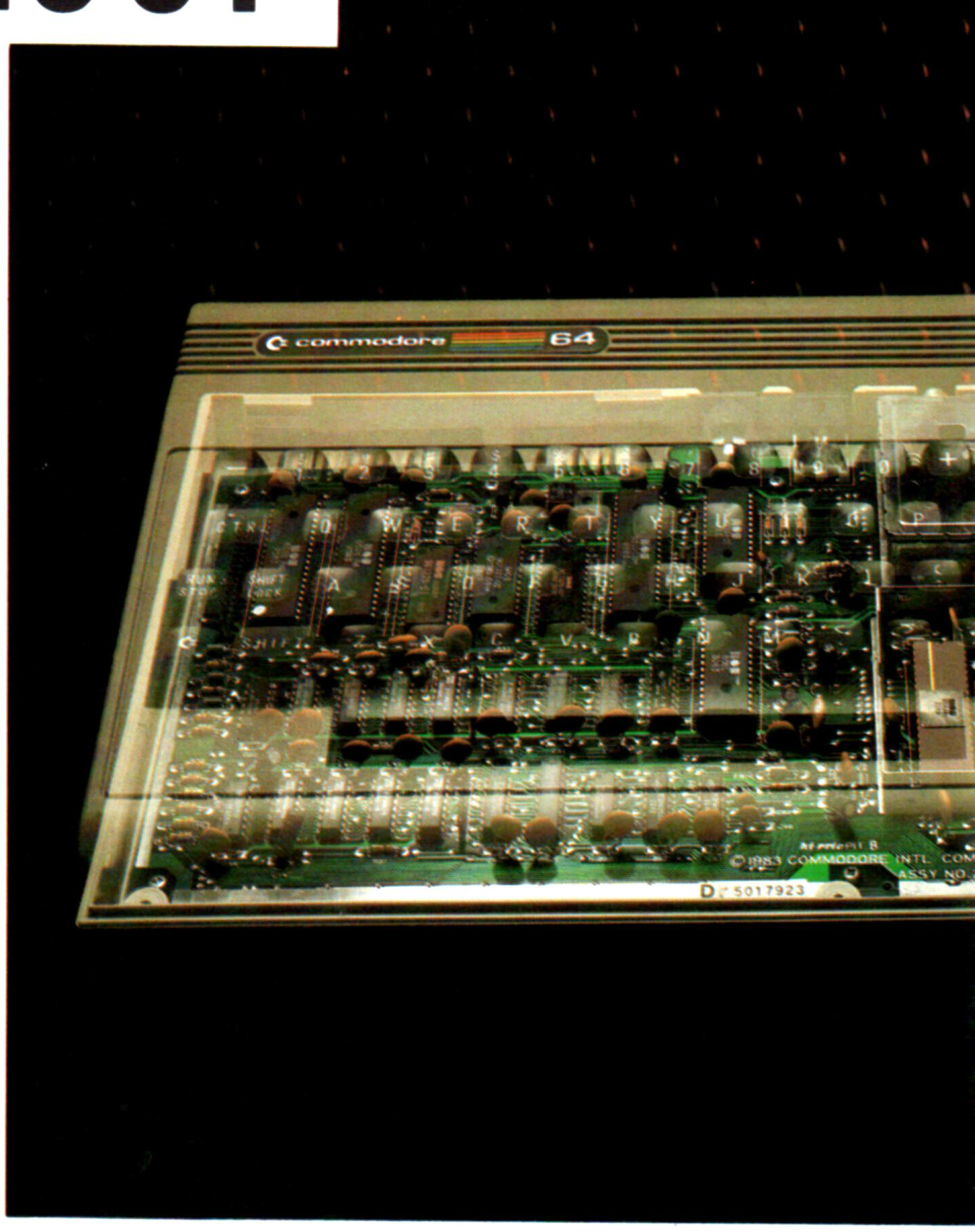
2) Construir una tabla de verdad para esta situación: "Se puede cargar un programa en un ordenador si se dispone de una grabadora de cassette 0 (OR) de una unidad de disco Y (AND) si el programa NO (NOT) está escrito para ser ejecutado en otra máquina".

3) Construir una tabla de verdad para el circuito:



Commodore 64

Máquina ideal para los amantes de los ordenadores personales, también se puede utilizar para aplicaciones de pequeñas empresas



El parecido físico entre el Commodore 64 y el Vic-20 es engañoso. Aunque entre ambos existe cierta compatibilidad de software, en cuanto al hardware el 64 significa un avance considerable. Comencemos por analizar los 64 Kbytes de RAM, configuración a la que alude el nombre de la máquina. Como oferta de mercado es una ventaja, ya que hasta el advenimiento del microprocesador de 16 bits, ésta era la mayor RAM de que disponían los microordenadores de gestión empresarial. No obstante, equipar un ordenador personal con tal cantidad de memoria entraña ciertas dificultades. Aunque un microprocesador de ocho bits como el 6502, que tanto se utiliza, puede direccionar un total de 64 Kbytes, en éstos se incluyen, además de la RAM, la totalidad de la ROM, y también los chips de entrada-salida con objeto de controlar el teclado, la pantalla y los periféricos.

La solución estaba en el *conmutador de banco*, una técnica con la que las secciones de memoria se encienden y se apagan en el mapa direccionable de memoria a medida que se las necesita. No existe ningún límite teórico en cuanto a la cantidad total de memoria que un ordenador puede incorporar utilizando este método, pero dado que el microprocesador aún puede direccionar sólo 64 Kbytes a la vez, cuanta más memoria haya, más conmutación de bancos se necesitará, como consecuencia de lo cual se reducirá la eficacia.

Caja de sorpresas

El SX-64 es una versión autónoma y portátil del Commodore 64 y se puede adquirir con diversas configuraciones. La versión más popular incorpora una unidad de disco (el espacio superior se puede utilizar para almacenar diskettes) y un monitor en color de cinco pulgadas. El SX-64 puede ejecutar, sin ninguna modificación, software basado en cartucho o en disco para el Commodore 64 estándar.

En muchos sentidos se trata de uno de los mejores ordenadores "de maleta", término que se acuñó para distinguir a este tipo de ordenadores de las verdaderas máquinas portátiles, como el Epson HX-20 y el Tandy Modelo 100. El teclado incorpora teclas totalmente esculpidas con los letreros de gráficos inscritos en el frente, y se puede separar de la unidad principal. En la parte superior de la carcasa hay una ranura para albergar los cartuchos de ROM; cuando no está ocupada por ningún cartucho, la abertura se cubre con una tapa para evitar el polvo.

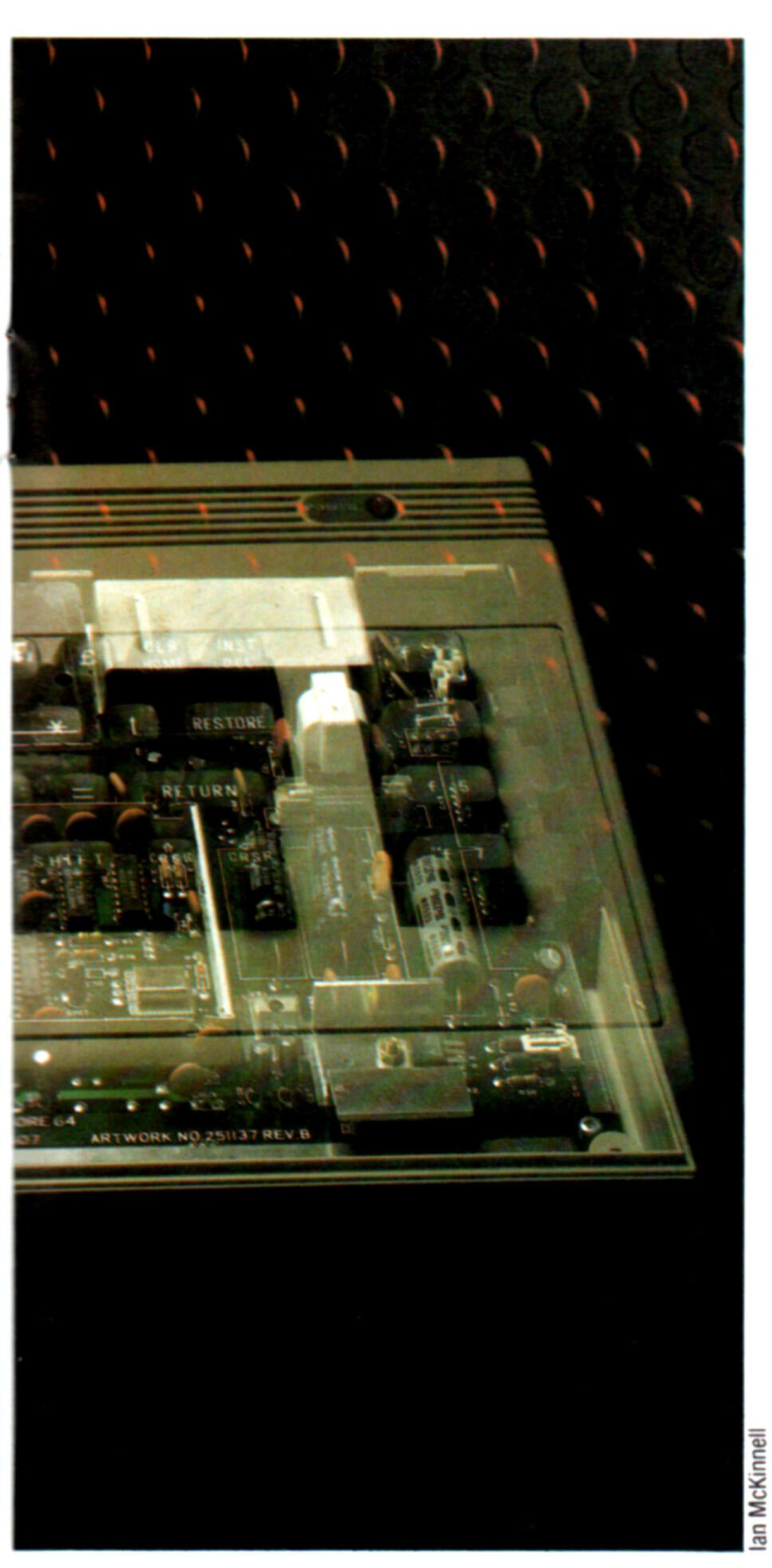
La carcasa es a la vez robusta y compacta y se parece al equipo portátil de comprobaciones que utilizan los ingenieros de reparaciones, en especial porque el asa para transportarla sirve al mismo tiempo como soporte. El asa tiene estrías para evitar que resbale sobre el escritorio, aunque el transportarla es algo incómodo. En líneas generales, el diseño físico es el mejor que ha producido la Commodore hasta la fecha, empañado sólo por el pequeño detalle del cable y el enchufe para la red eléctrica que no se pueden guardar en ningún sitio dentro de la carcasa



SX-64, cortesía de Commodore

Ian McKinnell





Ian McKinnell

En el caso concreto del Commodore 64, ello significa que si usted desea ejecutar un programa en BASIC, será necesario que se enciendan las ROM que contienen el intérprete del lenguaje, lo que reduce la cantidad de RAM disponible a 40 Kbytes (parte de los cuales aún se tendrán también que destinar a las variables del sistema y a la RAM de pantalla).

Aunque el conmutador de banco se ha incluido en muy pocos ordenadores personales con alguna modificación, en el Commodore 64 se obtiene mediante la utilización de un microprocesador especial. El 6510 es muy similar al 6502, de reconocida popularidad en el diseño de ordenadores personales. El conjunto de instrucciones es idéntico e incorpora un bus de datos de ocho bits, un bus de direcciones de 16 bits y diversas señales de control. Sin embargo, también ostenta una puerta programable de entrada-salida de ocho bits. Esto significa que en el chip hay ocho patillas adicionales, cada una de las cuales se puede establecer en 1 o 0, o se pueden utilizar para leer los valores colocados en ellas mediante un dispositivo externo. Normalmente este tipo de puertas se implementan mediante un chip especial (denominado PIO, PIA o VIA, según el fabricante), y un ordenador personal corriente incluye varios de éstos para manipular las puertas de los periféricos y el teclado.

A la puerta le corresponden en el mapa las dos posiciones de memoria más bajas ($0000 y $0001). La primera es para leer y escribir los bits individuales, mientras que la segunda indica si cada bit se establece como una entrada o como una salida. Tener esta puerta incorporada en el microprocesador significa que el 6510 sería ideal para incorporarlo a numerosos dispositivos domésticos, desde el lavavajillas hasta los juguetes programables. En el Commodore 64 se utiliza para seleccionar los bancos de memoria. Se podría hacer lo mismo con sentencias POKE en BASIC, pero actuando de este modo existe una clara posibilidad de "reventar" el sistema, lo que obligaría a tener que recomponer a continuación su ordenador. Por consiguiente, la mayoría de las conmutaciones de memoria se realizan en código de lenguaje máquina.

COMMODORE 64

MEDIDAS

404 × 216 × 75 mm

CPU

6510

MEMORIA

64 K de RAM, de los cuales hay 39 K disponibles para programas en BASIC.
20 K de ROM, incluyendo el generador de caracteres

PANTALLA

25 filas por 40 columnas. En baja resolución hay 16 colores disponibles desde teclado para caracteres, borde y fondo. El máximo en alta resolución es 320 × 200 pixels. Se pueden definir y utilizar hasta ocho sprites

INTERFACES

Palancas de mando (2) más lápiz óptico, RS232 (se necesita adaptador), en paralelo de 8 bits, cassette, en serie (para disco e impresora), monitor compuesto, entrada y salida de audio, TV, cartuchos

LENGUAJES DISPONIBLES

BASIC, FORTH, LOGO, lenguaje ensamblador 6502.

TECLADO

Estilo máquina de escribir, con teclas de cursor y cuatro teclas de función programable

DOCUMENTACION

El ordenador viene con un manual de instrucciones adecuado, pero para obtener el máximo provecho de sus configuraciones debe adquirir la *Guía de referencia para el programador*, o alguna de las muchas guías independientes que se han publicado para el Commodore 64

VENTAJAS

Mucha memoria estándar. Gráficos sprite. Sofisticado control de sonido. Teclado de calidad. Buena gama de periféricos. Dispone de más software de gestión que la mayoría de los ordenadores personales

INCONVENIENTES

Requiere la unidad para cassette del fabricante. BASIC pobre en órdenes útiles (a menos que compre un cartucho accesorio). Selección limitada de modalidades para gráficos y de resoluciones. Unidad de disco lenta

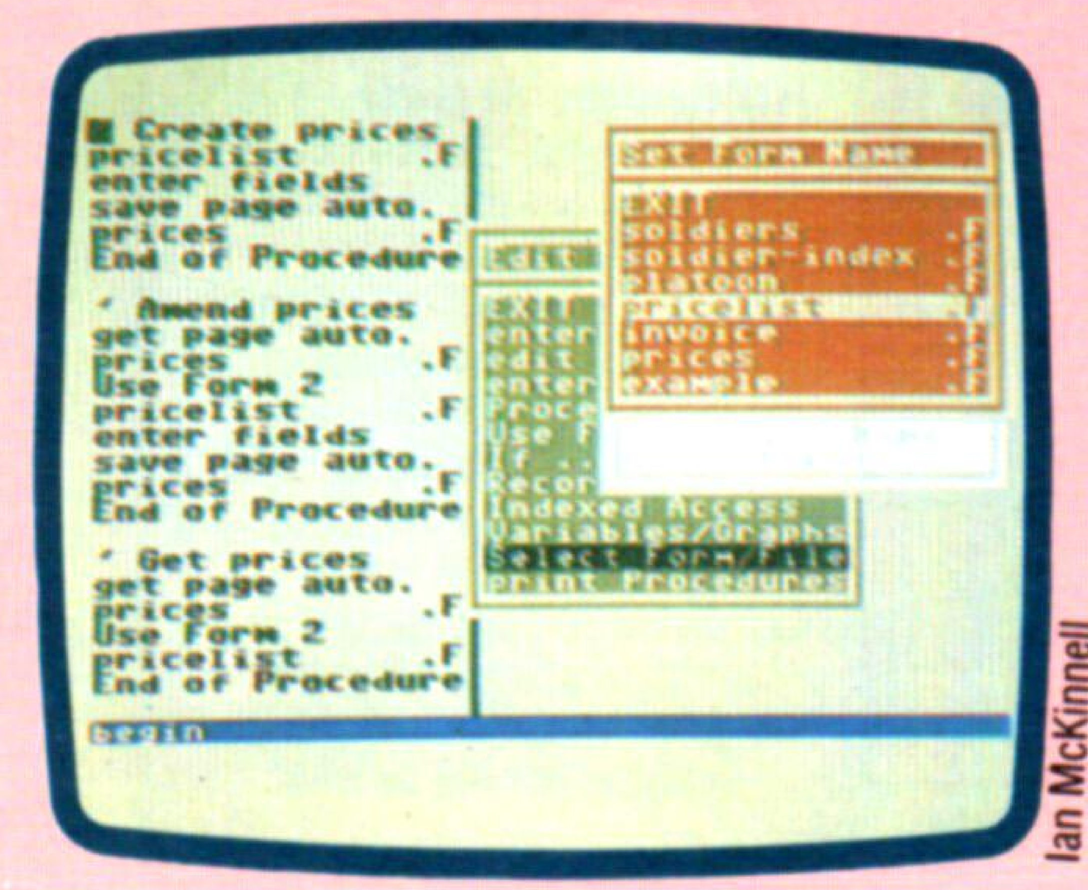
Posibilidades de gestión

Se puede afirmar que el Commodore 64 ha heredado de sus predecesores, el PET y el Vic-20, una gran proporción de su software de base. El intérprete de BASIC es muy parecido en las tres máquinas y también existen grandes similitudes en cuanto a los sistemas operativos de disco. Dado que el software de gestión desarrollado para la gama PET sólo se podía utilizar con máquinas Commodore, no es nada sorprendente que los productores de software fueran tan rápidos en sacar partido del nuevo mercado potencial que abrió el 64.

Para aplicaciones de gestión empresarial existe una amplia selección de paquetes para tratamiento de textos, varios de los cuales poseen verificadores de ortografía. Dos de los ejemplos más populares son el EasyWrite/EasySpell de Commodore y el VizaWrite/VizaSpell. Otros dos paquetes populares, aunque sin la opción de ortografía, son el Paperclip 64 y el Wordcraft 40. Este último es distinto de la mayoría de los paquetes de tratamiento de textos, por el hecho de que la pantalla visualiza el texto en el formato en el que éste se imprimirá finalmente, mientras que la mayoría de los otros visualizan "controles encajados", es decir, símbolos compuestos por un único carácter para significar un retorno del carro o que el título se ha de centrar en la página.

Existen hojas electrónicas y, entre ellas, un paquete que merece una mención especial: el CalcResult; es más caro que la mayoría de las hojas electrónicas para micros de precio económico, pero es a todo color, incluye una capacidad para visualizar gráficos de las cifras en cualquier columna de la hoja electrónica y trabaja en tres dimensiones. Todo ello quiere decir que se pueden retener simultáneamente en la memoria varias páginas de memoria y que se pueden sumar entre sí cifras de todas estas hojas.

Magpie es también un programa sobresaliente, que entra en la categoría de los generadores de aplicaciones. Una aplicación se define dibujando los trazados para los registros de pantalla y las formas impresas en la pantalla, y especificando luego las relaciones entre los campos en aquellos documentos:
IVA = TOTAL* 15 %, por ejemplo



Ian McKinnell


Ian McKinnell

Conexión cartuchos
Si se enchufa aquí un cartucho ROM (de hasta 16 Kbytes), anulará efectivamente cualquier otra memoria que ocupe las mismas posiciones. Si los primeros nueve bytes de la ROM contienen una secuencia específica de valores, entonces el programa "comenzará automáticamente" cuando se encienda. Así funcionan los cartuchos para juegos

Salida de TV
A diferencia del Vic-20, el Commodore 64 contiene un modulador de RF incorporado, de modo que la salida se puede conectar directamente a un televisor

Conexión audio-video
Se proporciona una señal de video compuesta para activar un monitor en color (aunque no un monitor RGB), y hay una salida de audio separada que se puede conectar con un sistema de alta fidelidad. También hay una línea de entrada de audio que le permite al usuario mezclar música grabada con sonidos sintetizados

Bus en serie
Ésta es una interface especial diseñada por la Commodore para activar varios dispositivos (incluyendo sus discos e impresoras) simultáneamente. El protocolo es similar al de la IEEE488 estándar, con la excepción de que sólo hay una línea de datos (en serie) en lugar de ocho líneas en paralelo

Conexión cassette
Todos los ordenadores Commodore requieren la unidad para cassette del fabricante. Cuando se comercializó por primera vez, el sistema Commodore era más rápido y más fiable que las unidades domésticas. Ahora sucede todo lo contrario

Conexión dispositivos
Esta conexión posee dos funciones. En primer lugar, puede recibir una interface en serie RS232 completa, si bien se requiere un accesorio para convertir los voltajes del 64 a los que utilizan la mayoría de los otros dispositivos en serie. Al mismo tiempo, puede funcionar como una puerta en paralelo que se puede utilizar para experimentación

Mapa de memoria
Los 64 K de espacio disponible de memoria se dividen en seis zonas, tres de las cuales normalmente están configuradas como RAM. Las otras tres contienen las ROM para el BASIC, el sistema operativo y los chips de E/S, pero para cada una de ellas existe un área de la RAM "en sombra" que se puede encender mediante el control del software. Sin embargo, esto sólo se puede conseguir cuando se utiliza el código de lenguaje máquina y no se necesita la ROM

32 K RAM
8 K RAM
8 K ROM de BASIC
4 K RAM
4 K RAM de E/S
8 K ROM de Kernal
RAM
RAM
RAM

Zona 1
Contiene permanentemente 32 K de RAM

Zona 2
Normalmente, 8 K de RAM. Si se insertara un cartucho ROM de 8 K, éste anularía a la RAM

Zona 3
Por lo general, 8 K de ROM, conteniendo el intérprete de BASIC. Se anularía al insertarse un cartucho de 16 K. Por otra parte, se pueden encender 8 K de RAM para reemplazar la ROM

Zona 4
4 K de RAM permanentemente

Zona 5
Suele contener todos los chips de E/S y algo de RAM. No obstante, bajo control del software aquí se puede encender el generador de caracteres basado en ROM

Zona 6
Al encenderse la máquina, contiene 8 K de ROM, incluyendo el Kernal, un sistema operativo mínimo diseñado para utilizarlo en todo el futuro hardware de Commodore. Cuando se utiliza código de lenguaje máquina se lo puede reemplazar por 8 K de RAM

Otros tres chips se encargan del resto de las configuraciones del 64. Hay un CIA *(Complex Interface Adaptor)* 6526, que es una versión más refinada de los PIA y VIA que hemos mencionado anteriormente. Además de las habituales líneas programables de entrada-salida, incluye sincronizadores y registros de desplazamientos para la conversión en paralelo de datos en serie y viceversa. Hay también un reloj de 24 horas con una alarma programable, que el intérprete de BASIC parece no utilizar en absoluto.

La visualización de gráficos y de video la manipula otro chip, el 6566, que es un nuevo desarrollo del *Video Interface Chip* (chip para interface de video), del cual el Commodore Vic-20 ha tomado su nombre. Éste ofrece distintas modalidades para las visualizaciones tanto de texto como de gráficos de alta resolución, y los gráficos sprite se han documentado muy bien. Aunque sólo puede manipular ocho sprites a la vez (frente a los 32 del Memotech MTX512, p. ej.), se pueden imitar bastantes más. Los sprites se definen en la memoria como un bloque de bytes y su posición se indica "colocando" (POKE) la dirección en los registros del chip Vic-II. Resulta relativamente sencillo conmutar el señalador rápida y repetidamente entre diferentes conjuntos de valores para simular de este modo más de ocho unidades.

El chip 6581 es conocido por SID (*Sound Interface Device*: dispositivo para interface de sonido), y contiene funciones mucho más avanzadas que algunos de los primeros sintetizadores de música diseñados especialmente. Además del control total de ADSR sobre la envoltura de volumen de cada sonido, las funciones que desempeña este chip incluyen la filtración, distintas formas de onda y modulación circular (es decir, la modificación de un sonido mediante otro).



Paso a paso

Para hablar con un ordenador hay que tener las ideas muy claras. La técnica de diagramación facilita esta tarea

Usted ya sabe en qué consiste un programa y las múltiples aplicaciones que tiene. Le falta por conocer en profundidad *cómo* se construyen, es decir, cuál es la lógica interna que sostiene a los programas. Una serie de capítulos que comienza con el presente le enseñará a analizar un problema y a visualizar los pasos que usted daría hasta llegar a la solución. Esta visualización mediante símbolos gráficos de uso universal es lo que se denomina *técnica de diagramación (flow-charting)*.

Cualquier problema, desde que se plantea hasta que se resuelve en forma de programa apto para introducirlo en el ordenador, debe pasar por una serie de fases que se podrían sintetizar en tres:

1. Análisis del problema y su resolución. Se entiende por análisis el estudio de los elementos que forman parte del planteamiento, así como cuándo y de qué manera tales elementos intervienen en lo que acaba siendo una solución lógica del problema, bien sea mediante un desarrollo puramente mental o con la utilización de notas y apuntes. La construcción se basa en un conjunto de ideas que, secuencialmente seguidas, llevan a una solución.
2. Diagrama: representación gráfica del análisis.
3. Codificación. Obtiene el programa, es decir, transcribe el análisis en el lenguaje informático usado (BASIC, PASCAL, etc.).

Esta tercera fase depende siempre de las dos previas. Sin duda la primera fase (el análisis) es completamente imprescindible, pues constituye la solución lógica que sólo la mente puede elaborar, mientras que la segunda facilita la programación.

La diagramación podría definirse como el medio que facilita el análisis de las aplicaciones mediante figuras geométricas simbólicas que representan las diferentes fases de la solución de un problema.

Se usan estos diseños gráficos porque resultan más claros y simples a la hora de un seguimiento que si la resolución se describiera a base de referir todos los pasos con un texto convencional.

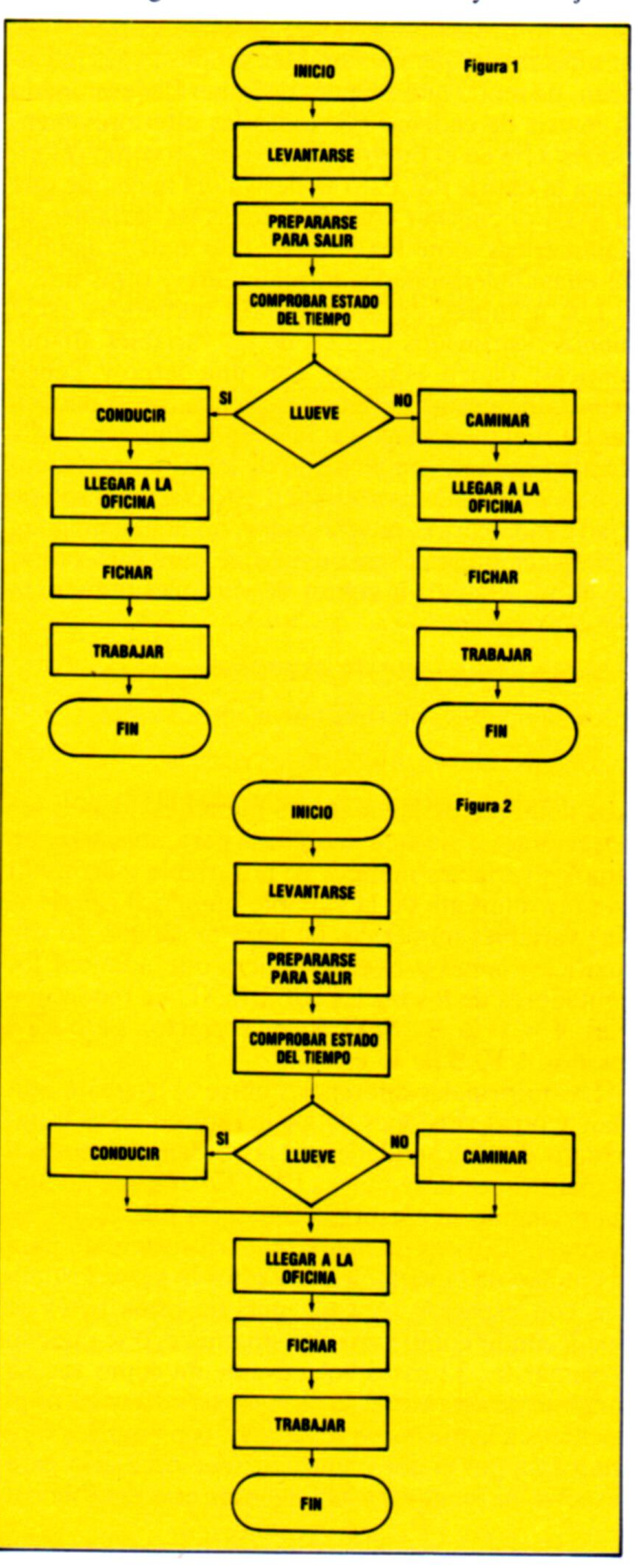
Todo proceso encaminado a resolver un problema se compone de una secuencia determinada de fases elementales, simbolizables mediante diagramas por complejo que sea el problema.

La figura 1 es la representación gráfica de este sencillo ejemplo: "Juan realiza cada mañana los siguientes actos: se levanta, se prepara para salir y comprueba qué tiempo hace. Si llueve, va en coche hasta la oficina, llega, ficha y comienza a trabajar. Mientras que si el tiempo es bueno camina hasta la oficina, donde ficha y se pone a trabajar".

Cada acto o fase se representa por medio de un rectángulo, mientras que la fase decisoria del estado del tiempo se hace con un rombo (también podría ser un hexágono). Todos ellos van unidos por líneas que marcan el orden de la secuencia lógica.

Ahora bien, frente a un mismo problema podemos llegar a diferentes conclusiones a la hora de presentar una solución. Si todas son buenas, ¿cuál escogeremos? Siempre resulta más válida aquella en la que no se vuelven a escribir unas mismas partes del programa, denominadas *rutinas*, comunes a varios caminos fundamentales.

Así, la figura 2 ilustra cómo se puede evitar la repetición del área en que se representan los elementos "llegar a la oficina", "fichar" y "trabajar".



El Basic ZX de Sinclair

El BASIC se ha convertido en el lenguaje oficial de los microordenadores, pero casi todos poseen su propia versión. Una de las más utilizadas es el Basic ZX de Sinclair

Empecemos por los nombres de las variables, que siempre constituyen una fuente de confusión entre los dialectos del BASIC. En el Sinclair, los nombres de las variables alfanuméricas deben tener una sola letra y no hay distinción entre letras en mayúscula y letras en minúscula. Esto significa que las variables a$ y A$ se refieren a la misma posición de memoria. Los nombres de las matrices de cadenas siguen las mismas reglas que las variables simples, y las inicializan, de modo que después de haber DIMensionado la matriz de cadenas H$, todas las ulteriores menciones a H$ en el programa se tomarán como referidas a la matriz H$. Esto se deriva del hecho de que el BASIC de Sinclair considera todas las variables alfanuméricas como variables de tipo matriz, algunas de ellas DIMensionadas formalmente y otras no.

Los nombres de las variables numéricas están menos restringidos que los de las variables alfanuméricas: deben empezar con una letra y deben estar compuestos de letras o dígitos, pero pueden ser de cualquier longitud. Pueden incluir espacios y pueden ser una combinación de letras en mayúscula y en minúscula, pero si bien estos factores son de gran ayuda para el programador, no tienen ninguna significación para la máquina, que hará caso omiso de ellos. Algunos nombres de variables numéricas posibles serían:

```
qwert, ub40, curso de informática
```

y los siguientes son sus equivalentes exactos:

```
QWERT, UB 40, Curso de Informática
```

Los nombres de las matrices numéricas deben ser letras únicas, sin que sea óbice para utilizarlas en simples variables numéricas: la variable matriz v(8) es muy diferente de la variable numérica simple v. Las variables numéricas de letra única que no son matrices, como v, son las únicas que admiten los contadores de los bucles FOR...NEXT, de modo que FOR V = 1 to 9...NEXT V es correcto, pero FOR bucle = 1 TO 9 no lo es.

Las principales diferencias entre el lenguaje Sinclair y otras versiones de BASIC radican en el tratamiento de las cantidades de las cadenas. Vamos a comenzar por la sentencia DIM. En el BASIC de Sinclair, cuando se ejecuta la sentencia DIM a$(12), se reservan 12 bytes de memoria exclusivamente para el empleo de la variable a$, y estos bytes se inicializan con espacios. A cada uno de estos bytes se puede aludir como variables subíndice, o se pueden reclamar los 12 bytes colectivamente como a$. La longitud de esta variable siempre será 12, y las asignaciones a la misma se rellenarán con espacios o se truncarán por la derecha cuanto sea necesario para respetar su longitud. Supongamos que escribimos:

```
DIM a$(12): LET a$ = "123456789"
```

Entonces a$ realmente contendrá estos nueve caracteres "123456789" seguidos de tres espacios, dando un total de 12 caracteres. Si, en cambio, escribimos:

```
DIM a$(12): LET a$ = "ABCDEFGHIJKLMN"
```

Entonces a$ en realidad contendrá sólo los primeros 12 caracteres "ABCDEFGHIJKL": la cantidad de la cadena "ABCDEFGHIJKLMN" se ha truncado por la derecha para que encaje en la longitud DIMensionada de a$. Si ahora escribimos:

```
LET a$(2 TO 5) = "1234"
```

entonces a$ contendrá "A1234FGHIJKL". Esto ilustra la eficacia de Sinclair en la manipulación de cadenas: todas las cadenas son tratadas como matrices de cadenas de dimensión simple, las matrices pueden llevar o no subíndice, y se puede acceder a los elementos individuales de una matriz (de forma individual o como parte de una subcadena) mediante subíndices. Asimismo, ilustra otra gran diferencia respecto a otras versiones de BASIC. En otros sitios, DIM a$(12) crea 12 variables alfanuméricas separadas denominadas a$(1), a$(2), etc., cada una de las cuales tiene la longitud de la expresión a ella asignada. Si a una variable alfanumérica determinada no se le hubiera asignado nada, entonces su longitud sería 0 y contendría sólo la cadena nula, "".

En otras versiones de BASIC esta forma de manipular las cadenas requiere de las diversas funciones para cadenas, LEFT$, RIGHT$, MID$ y algunas veces INSTR, para posibilitar la manipulación de subcadenas y la partición de cadenas de la forma dicha. Pero esto no es así en el BASIC de Sinclair. Los equivalentes Sinclair de estas funciones para cadenas son:

```
LEFT$(A$,N) = A$(TO N)
```

(que significa los N caracteres más a la izquierda de A$):

```
RIGHT$(A$,N) = A$(LEN A$-N+ TO)
```

(que significa los N caracteres más a la derecha de A$); y

```
MID$(A$,P,N) = A$(P TO P+N-1)
```

(que significa los N caracteres de A$ desde la posición P hacia adelante). Por su parte, la función:

```
LET S = INSTR(A$,"teststring")
```

(que significa hallar la posición de comienzo en A$ de la subcadena cuyo valor sea, por ejemplo, "teststring") se puede reemplazar por:

```
LET Y$ = A$:LET Z$ = "teststring":GOSUB
   9900:LET S = POSN
```



```
9900 LET    ZL = LEN    Z$:LET    SL = LEN
  Y$-ZL+1:LET POSN = 0
9910 FOR K = 1 TO SL
9920 IF  Y$(K  TO  K+ZL-1) = Z$  THEN  LET
  POSN = K:LET K = SL
9930 NEXT K:RETURN
```

Observe cómo la variable alfanumérica Y$ es tratada como una variable subíndice de tipo matriz, aun cuando no se la haya DIMensionado. Dado que en BASIC de Sinclair todas las variables alfanuméricas son variables de tipo matriz, una variable alfanumérica que no esté DIMensionada es implícitamente una matriz de caracteres únicos, de dimensión única y longitud variable; de estar DIMensionada, la longitud de su elemento está determinada por el último número de la sentencia DIM. Mientras que en otras versiones de BASIC DIM y$(8,7) crea una matriz de dos dimensiones, en Sinclair crea una matriz de dimensión única de ocho elementos, cada uno de los cuales tiene una longitud fija de siete caracteres.

La estricta atención que el BASIC de Sinclair presta a la longitud de las variables alfanuméricas DIMensionadas significa que sentencias aparentemente sencillas pueden producir efectos diferentes, según se haya ejecutado o no una sentencia DIM. Si a$ es una variable alfanumérica simple, entonces LET a$ = "" hace que el contenido de a$ sea igual a la cadena nula ("") y la longitud de a$ sea igual a cero. No obstante, si previamente se hubiera ejecutado DIM a$(7), entonces LET a$ = "" hace que el contenido de a$ sea igual a siete espacios, y la longitud de a$, igual a siete (que tendrá siempre, a partir de la sentencia DIM). Además, en dicho caso, aun cuando se hubiera ejecutado LET a$ = "", una condición como:

```
IF a$ = "" THEN PRINT "cadena nula"
```

fracasará, y no se imprimirá nada: a$ es igual a siete espacios y no a la cadena nula.

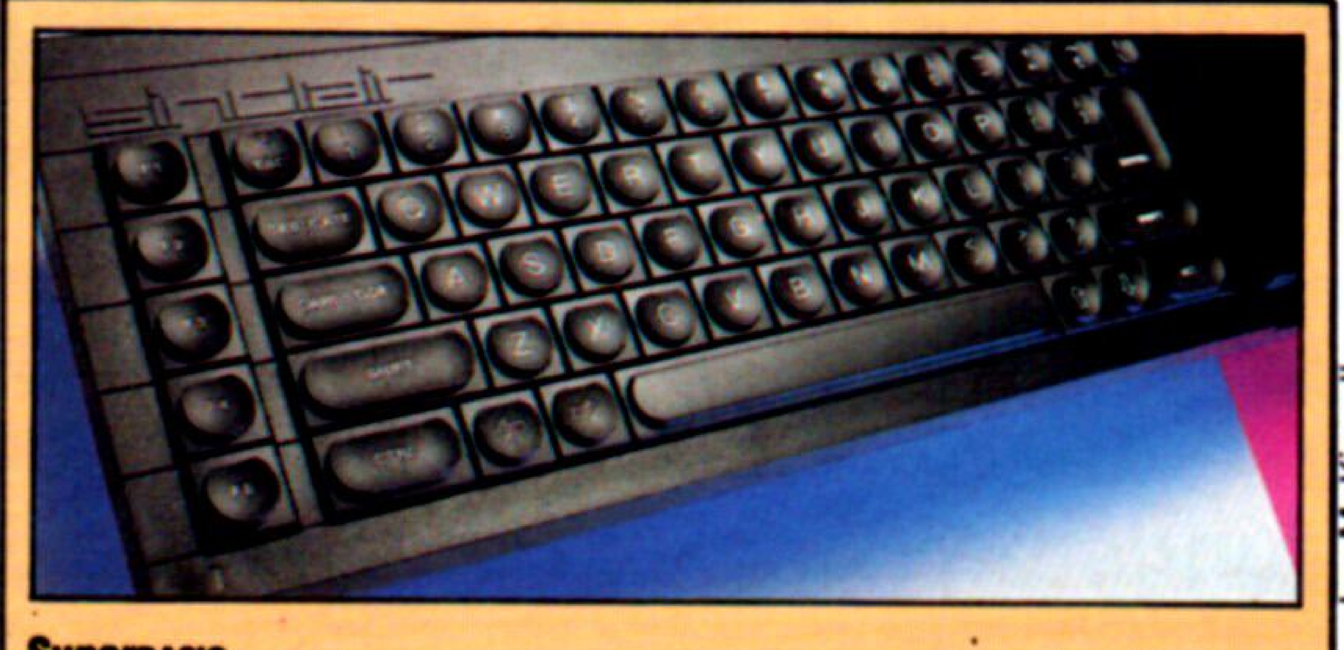
Si necesita verificar de esta manera los elementos de la matriz de cadenas, entonces probablemente sea mejor reservar con este fin una variable alfanumérica, DIMensionarla a la longitud de la variable de matriz más larga utilizada en el programa, y verificar sus variables de matriz mediante ella, así:

```
100 DIM a$(12,34)
120 DIM b$(7,56)
140 DIM N$(56)
150 REM N$ se utilizará como la cadena vacía
.........................................
580 IF b$(3) = N$(TO 56) THEN PRINT "vacía"
590 IF a$(11) = N$(TO 34) THEN PRINT "vacía"
.........................................
```

Aquí N$ sólo se emplea como la cadena vacía y, si no se utilizara de esta manera, entonces las condiciones de las líneas 580-590 habrían de echar mano de literales, de modo que:

```
580 IF b$(3) = "          " THEN PRINT "vacía"
585 REM 56 espacios entre las comillas
```

Esto es incómodo y es susceptible de error. Una alternativa a la utilización de N$ de esta forma consiste en DIMensionar todas las variables de matriz con un elemento más del que se necesita y utilizar ese último elemento como una cadena vacía para las condiciones de dicha matriz, de modo que la línea 590 podría decir:



Ian McKinnell

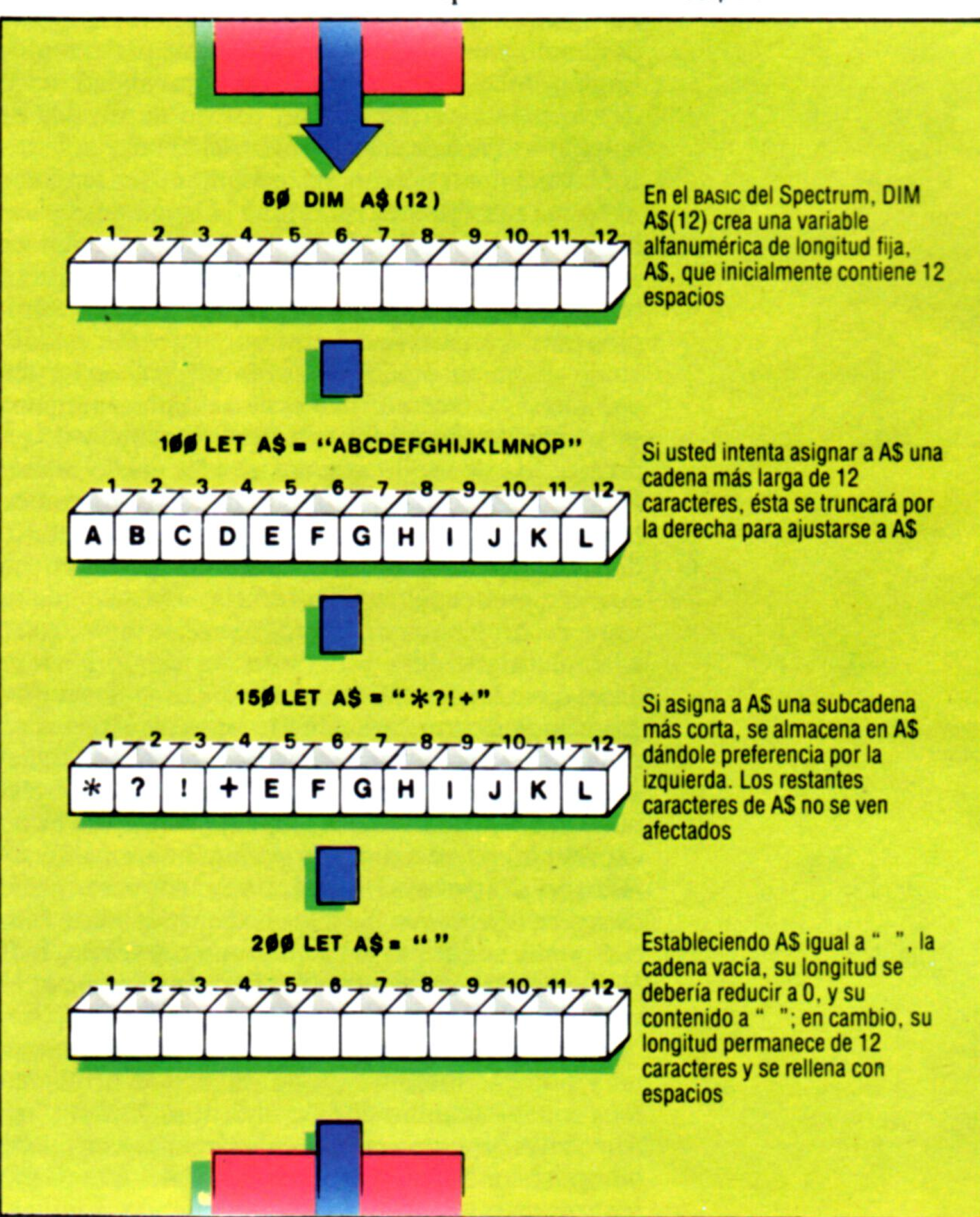
SuperBASIC
El SuperBASIC de Sinclair posee una gama de órdenes considerablemente mejorada respecto al BASIC ZX, pero la novedad más llamativa es el abandono del sistema de entrada de palabras clave tecla a tecla, común al ZX80, al ZX81 y al Spectrum. Originalmente se pensó como una medida de economía para los usuarios (se creyó que pulsar una sola tecla en lugar de digitar una palabra entera resultaría estimulante). El sistema fue provisto de una variedad de "modalidades" diferentes para permitir que la entrada de caracteres únicos se diferenciara de la entrada de palabras clave. Este sistema fue atractivo para aquellos usuarios de Sinclair que nunca antes se habían enfrentado a un teclado, pero resultó frustrante para quienes ya habían utilizado una máquina de escribir

```
590 IF a$(11) = a$(12) THEN PRINT "vacía"
```

dando por sentado que a$(12) no se emplea nunca y que, por consiguiente, sólo contiene espacios.

Por último, observe que en el Sinclair el primer elemento de cualquier matriz posee el subíndice uno, mientras que en algunas otras versiones de BASIC el primer elemento de una matriz tiene el subíndice cero. En el próximo capítulo del curso terminaremos este análisis del BASIC Spectrum.

Cadenas de Procusto
El personaje mitológico griego Procusto era un mesonero que tenía camas de un solo tamaño y estiraba o acortaba a sus huéspedes para que se adaptaran a ellas



En el BASIC del Spectrum, DIM A$(12) crea una variable alfanumérica de longitud fija, A$, que inicialmente contiene 12 espacios

Si usted intenta asignar a A$ una cadena más larga de 12 caracteres, ésta se truncará por la derecha para ajustarse a A$

Si asigna a A$ una subcadena más corta, se almacena en A$ dándole preferencia por la izquierda. Los restantes caracteres de A$ no se ven afectados

Estableciendo A$ igual a " ", la cadena vacía, su longitud se debería reducir a 0, y su contenido a " "; en cambio, su longitud permanece de 12 caracteres y se rellena con espacios

Kevin Jones

Primeros conceptos

Este primer capítulo del curso de programación en lenguaje máquina pretende hacer perfectamente comprensibles los fundamentos de la programación de ordenadores

La programación en lenguaje máquina es la llave que deja al descubierto la verdadera potencia del microprocesador. De este modo, el programador tiene en sus manos el control directo de la máquina.

El lenguaje máquina tiene frases como ésta:

```
INSTK: SBC $D9FA, X; Outport valor bandera
```

o ésta:

```
DE23 FD FA D9
```

o ésta:

```
11011110 00100011 11111101 11111010 11011001
```

Algunas veces se escribe así:

```
1240 LET ACC = ACC—FLAG (X)
```

Y otras veces de esta manera:

```
PERFORM FLAG—ADJUST THROUGH LOOP 1
```

Son frases de un mismo código y, puesto que está destinado a una máquina informática, se denomina *lenguaje máquina*. A la máquina, en realidad, no le dice más que se trata de un patrón de niveles de voltaje o una corriente de electricidad.

Normalmente, cuando hablamos de lenguaje máquina nos estamos refiriendo al lenguaje ensamblador y el primer ejemplo que hemos ofrecido en este artículo es una instrucción en lenguaje ensamblador 6502. El que hayamos añadido otros ejemplos más era para demostrar que no existe un lenguaje máquina específico como tal, sino un buen puñado de diferentes maneras de representar una secuencia de acontecimientos electrónicos, y de representarla de modo que nos resulte más o menos fácil de comprender. De modo que lo primero que hay que aprender acerca del lenguaje máquina (o del lenguaje ensamblador, pues de momento no nos va a preocupar su diferencia) es tan sólo que se trata de un lenguaje de programación como otro cualquiera. Pero la programación siempre viene antes que el lenguaje: tanto si usted escribe sus programas en ensamblador IBM, en BASIC Atari o en cualquier otro lenguaje, antes de sentarse al teclado debe haber resuelto el problema de la programación en su cerebro. El lenguaje en el cual usted exprese luego su solución, obviamente incidirá en la forma del programa final. En realidad, se puede escoger entre varios posibles lenguajes porque buscan que la codificación de su programa resulte más fácil, más corta o más legible. Pero la solución es lo primero: el contenido es antes que la forma.

En este caso, ¿por qué llamarlo lenguaje máquina y por qué molestarnos en utilizarlo? Lo llamamos con ese nombre porque el conjunto de sus instrucciones se corresponde exactamente con el de las operaciones "primitivas" o esenciales que puede realizar un microprocesador determinado. Empleamos el código de lenguaje máquina cuando importa dirigir la operación del microprocesador paso a paso, en vez de que un intérprete de lenguaje de programa lo controle de forma más general.

La razón más común por la que se utiliza es la velocidad: si su programa direcciona al procesador de forma más o menos directa, entonces evita el relativamente largo camino de la traducción del programa. En otras palabras, eliminando el intermediario se gana tiempo. Es decir, tiempo de ejecución del programa. La detallada codificación, verificación, depuración, modificación y mantenimiento de un programa en lenguaje máquina es probable que nos pida al menos el doble de tiempo que esas mismas operaciones necesitan en un lenguaje de alto nivel. El carácter oscuro y casi ininteligible del lenguaje máquina estimuló el desarrollo de lenguajes como el COBOL y el BASIC.

Si el conjunto de instrucciones en lenguaje máquina es el de las operaciones del procesador, ¿entonces qué son estas operaciones y qué es lo que hace el procesador? En términos lo más sencillos posible, la unidad central de proceso (CPU: *Central Processing Unit*) de un ordenador es un interruptor que controla el flujo de corriente de un sistema informático a través de los componentes de dicho sistema. Estos componentes son la memoria, la unidad aritmético lógica (ALU: *Arithmetic Logic Unit*) y los dispositivos de entrada-salida. Cuando usted pulsa una tecla está dando entrada a alguna información; en la máquina, sin embargo, simplemente está generando un patrón de voltajes en la unidad de teclado. La CPU conmuta el patrón desde el teclado a parte de la memoria, luego conmuta un patrón correspondiente desde algún lugar de la memoria hasta la pantalla de modo que en ella aparezca un patrón de caracteres. Puede que este proceso le resulte parecido al de una máquina de escribir; pero en ésta hay una conexión mecánica entre pulsar una tecla e imprimir un carácter, mientras que en un ordenador esa conexión sólo existe en virtud de que la CPU conmuta los patrones de voltaje correctos de un lugar a otro. Algunas veces pulsar una tecla no produce en la pantalla un carácter: la pulsación de una tecla puede destruir un asteroide, o guardar un programa, o eliminar un archivo en disco, además de imprimir una letra. La operación depende de cómo y dónde conmute la CPU la corriente eléctrica.

En este análisis esquemático, la CPU está situada en el corazón del sistema y toda la información (o corriente eléctrica) debe pasar a través de ella desde un componente a otro. En realidad, la CPU y el sistema son algo más complicados, pero no es un análisis engañoso. Puede imaginar la CPU como un controlador maestro que establece interruptores menores a través de todo el sistema para controlar



el flujo de electricidad y que, por consiguiente, controla indirectamente el flujo de información, en vez de imaginar que canaliza toda la información físicamente a través de sí misma.

Los efectos de las operaciones de conmutación de la CPU se pueden clasificar, para nuestros fines, como: operaciones aritméticas y operaciones lógicas, operaciones de memoria y operaciones de control. Todas estas operaciones son resultado de conmutar la información a través de distintos caminos del sistema y de la CPU, y para la CPU es como si todas ellas se trataran de la misma cosa.

Las operaciones aritméticas son, en realidad, la configuración más importante de la máquina. La CPU puede sumar dos números entre sí o restarle uno al otro. La resta se consigue representando uno de los números como un número negativo y sumándole ese número negativo al otro número; 7 + 5 = 12 en realidad significa:

(más 7) sumado a (más 5) es igual a (más 12).

7 − 5 = 2 en realidad significa:

(más 7) sumado a (menos 5) es igual a (más 2).

La multiplicación y la división se consideran como sumas o restas repetitivas, de modo que se puede programar a la CPU para que también imite estos procesos. Si la CPU puede hacer frente a las cuatro operaciones aritméticas, entonces puede hacer frente a cualquier proceso matemático. Merece la pena recalcar, no obstante, que todo su potencial matemático descansa en la simple capacidad de sumar dos números.

Para los fines que perseguimos actualmente, las operaciones lógicas se pueden describir como la capacidad de comparar dos números: no sólo en cuanto al tamaño relativo, sino también respecto al patrón de sus dígitos. Es fácil ver que siete es mayor que cinco, porque podemos quitarle cinco a siete y aún tendríamos un resultado positivo. La CPU posee la capacidad de hacer esa clase de comparación, y también puede comparar 189 con 102 y reconocer que ambos números tienen el mismo dígito en la columna de las centenas.

Esencialmente, la CPU puede realizar dos operaciones de memoria: puede copiar información de una posición de memoria en su propia memoria interna, y puede copiar información de su memoria interna a otra posición de memoria. Haciendo estas dos cosas, una después de la otra, puede, por consiguiente, copiar información de cualquier parte de la memoria a cualquier otra parte de ésta. Para que la memoria nos sea útil, la CPU debe ser capaz de hacer estas dos cosas, y estas dos operaciones son todo cuanto necesita para conseguir una administración completa de la memoria.

Las operaciones de control son, en realidad, decisiones acerca de la secuencia en la cual la CPU realiza las otras operaciones que hemos descrito aquí. Por el momento no es importante entenderlas con mayor profundidad: si uno acepta que la CPU puede tomar decisiones relativas a su propia operación, eso ya es suficiente por ahora.

De modo que la CPU puede realizar operaciones aritméticas, comparar números, desplazar la información por la memoria y decidir su propia secuencia de operaciones. Ésta es una lista sencilla de procedimientos y, así y todo, ¡describe y especifica completamente una máquina de informática ideal! Si la CPU puede hacer estas cuatro cosas, haciéndolas en la secuencia correcta puede realizar cualquier trabajo informatizable. La secuencia correcta, por supuesto, es el programa de ordenador para el trabajo determinado y aquí es donde entramos nosotros como programadores. Si la CPU pudiera generar sus propias secuencias de operaciones, no se nos necesitaría a nosotros para nada.

Puede que usted no esté aún muy convencido de que los cuatro tipos de operaciones que hemos descrito sean la idea cabal de un ordenador, así que vamos a tomar un programa en BASIC y reducirlo a las operaciones generales efectuadas. ¿Qué son estas operaciones fundamentales? En cualquier programa se encuentra con variables, que son simplemente los nombres de los lugares de la memoria en donde está almacenada la información. La mayoría de los programas realizan alguna clase de operación aritmética con algunas de estas variables. Una vez efectuada la operación, a menudo se compararán dos informaciones y, como resultado, se ejecutará un conjunto de instrucciones u otro. Finalmente la información suele entrar en el programa del usuario por el teclado, y salir hacia el usuario a través de la pantalla.

Salvo la sentencia relativa a entrada y salida, esta descripción no contiene más que las cuatro operaciones elementales de la CPU expresadas con diferentes palabras. Y, si se acepta por el momento que para la CPU todos los dispositivos de entrada-salida son sólo zonas especiales de la memoria, entonces la imagen del ordenador ideal ejecutando programas reales es completa. Por consiguiente, la ejecución de un programa se puede describir como un flujo de información dirigido hacia adentro, alrededor y hacia afuera del ordenador; usted proporciona alguna información mediante el teclado, esa información la manipula su programa, y en la pantalla aparece otra información.

Si el ordenador idealizado no es más que una CPU y algo de memoria, entonces antes de seguir adelante debemos analizar la memoria del ordenador: ¿qué es y cómo funciona?

Imagínese un circuito eléctrico sencillo que conste de una pila, un interruptor y una bombilla: si se acciona el interruptor se enciende la luz, y permanece encendida hasta que se agota la pila o hasta que se cierra el interruptor. Entonces el estado de la bombilla (ON u OFF: encendida o apagada) es una información, y todo el circuito es un dispositivo de memoria que registra esa información. Supongamos ahora que el interruptor está localizado a la entrada de una fábrica y que la luz está situada en el despacho del director. Cuando el primer empleado llega a la fábrica, acciona el interruptor de la entrada y el director, al ver la luz encendida en su despacho, sabe que alguien ha llegado al trabajo. El director no necesita estar en el despacho cuando la luz se enciende: puede mirar la bombilla en cualquier momento para averiguar si alguien ha llegado. La información de que alguien se ha presentado a trabajar está almacenada en el circuito.

Así es casi exactamente como la información se almacena en la memoria del ordenador: toda información se reduce a la presencia o a la ausencia de electricidad en un circuito. Naturalmente, no se trata tan sólo de esto, de manera que vamos a mejorar el sistema de información de la dirección de la fábrica. Supongamos que tenemos cuatro cir-

cuitos interruptor-bombilla separados (los cuatro interruptores en una fila junto a la puerta y las cuatro bombillas en una fila correspondiente en el despacho), de tal forma que cerrando el interruptor situado más a la izquierda se ilumine la bombilla localizada más a la izquierda, y así sucesivamente. Ahora imaginemos que a cada empleado se le indica que conecte los interruptores de una manera distinta a la de su compañero, de modo que cuando llegue Margarita conectará el primero y el segundo interruptor y dejará abiertos el tercero y el cuarto; que Gerardo cierre el cuarto interruptor y abra los otros tres; que Pablo cierre el primero y el tercero y abra el segundo y el cuarto; y así para los 16 empleados. Las bombillas informarán ahora al director cuál es el empleado que ha llegado al trabajo.

Supongamos que la posición OFF de cada interruptor está etiquetada 0 y que la posición ON está etiquetada 1: más claramente diremos que Margarita tiene que establecer el panel de interruptores en 1100 (los dos primeros interruptores ON, el tercero y el cuarto OFF); a Gerardo le corresponde el patrón 0001 (el cuarto interruptor ON, los otros tres OFF) y Pablo ha de establecerlos en 1010 (el primero y el tercero ON, los otros dos OFF). Si el director entiende el 1 como bombilla encendida y el 0 como apagada, los empleados y él estarán hablando el mismo lenguaje de identificación. Para todos ellos, "0001" significa Gerardo.

¿De cuántas combinaciones diferentes de interruptores disponemos? Cada interruptor puede estar en una de dos posiciones y hay cuatro interruptores, de modo que hay 2 × 2 × 2 × 2 = 16 posibilidades diferentes, que serían:

0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111,
1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111

Observe con qué rapidez hemos pasado de la imagen concreta de las bombillas en una habitación a la región abstracta de patrones de unos y ceros. Sólo con un poco más de abstracción podremos convertir estos patrones en números.

Trate de imaginar y escribir los números a medida que los va contando. Usted puede escribir de cero a nueve muy fácilmente, porque cada uno de estos números tiene un nombre y un símbolo distintos para representarlo. Pero ¿qué es lo que escribe después de nueve? Dispone de un nombre (diez) para ese número, pero no tiene un rasgo nuevo que lo represente. Inconscientemente echa mano de dos símbolos que usted ya utilizó por separado, el 1 y el 0, y así ha compuesto 10, un símbolo arbitrario para el diez. Lo mismo sucede con el once, pues en realidad usted escribe dos unos, el trece, catorce, etc., hasta agotar con el noventa y nueve, 99, las combinaciones por parejas de los únicos diez símbolos de que dispone (excluyendo las que tendrían el cero delante). Para representar cien, no tiene más remedio que pasar de las parejas de símbolos a los tríos: 100 (uno-cero-cero), 110 (uno-diez), etc. Esto parece trivial, pero quizá recuerde todavía lo difícil que le resultó aprenderlo en la escuela. Sin usar aquellas hojas cuadriculadas que le separaban las centenas de las decenas y de las unidades, ahora instintivamente interpreta el número 152 compuesto por una centena, cinco decenas y dos unidades, o lo que es lo mismo: 1 × 100 + 5 × 10 + 2 = ciento y cincuenta y dos.

¿Cómo escribiría usted los números si sólo le permitieran dos símbolos, el 0 y el 1? Podemos escribir fácilmente el 1, ¿pero cómo podemos representar el número siguiente? Nos hemos quedado sin dígitos exclusivos, de modo que debemos echar mano de los que nos permiten (como hicimos al disponer de diez dígitos) y escribir el siguiente número, que es el dos, como 10. Desde ahora acordamos que el número llamado "dos", en este sistema de tan sólo dos símbolos (binario) se escribirá 10. Seguimos contando y el número siguiente es tres, que lo debemos escribir como 11. ¿Y después, qué? Nos hemos quedado sin combinaciones de dos dígitos, de modo que el número siguiente, cuatro, se debe representar como 100; cinco debe ser 101 (cuatro y uno), seis es 110 (cuatro y dos) y siete es 111 (cuatro y dos y uno). En pp. 54 y 55 tenemos la expresión gráfica de lo que estamos tratando.

Así como un número decimal como 152 significa: (1×100) + (5×10) + (2×1), el número binario 101 significa: (1×4) + (0×2) + (1×1). En vez de emplear columnas de centenas, decenas y unidades para nuestros números, debemos utilizar columnas marcadas con cuatros, doses y unidades. En un número decimal el valor de un dígito se multiplica por diez para cada columna que se desplaza hacia la izquierda; en un número binario, el valor de un dígito se multiplica por dos para cada columna que se desplaza hacia la izquierda.

De modo que el sistema binario es sólo una forma diferente de representar los números, tan arbitraria como la decimal. Si conoce la numeración romana no le resulta difícil entender si escribimos que en *Blancanieves* hay VII enanitos; ¿entiende que se puede escribir 111 enanitos? El número verdadero de enanos no se altera en virtud de la forma en que lo representemos, pero es aconsejable leerlo así "uno uno uno binario" y escribirlo como "111 b", de modo que no se confunda con una representación decimal.

Ahora podemos volver a nuestra analogía original de cómo los empleados de la fábrica conmutan patrones, y decidirnos por un método que haga que éstos resulten más fáciles de utilizar. Lo más sensato que se puede hacer consiste en tratar a estos patrones como números binarios de cuatro dígitos. Esto significa que la señal de Margarita es 1100 binario, que en decimal es 12. La señal de Gerardo es 0001 binario (1 decimal), y la señal de Pablo es 1010 binario (10 decimal). Cuando el director observa el patrón de luces en su despacho, él lo puede leer como un número binario, convertirlo a su equivalente decimal y consultar la lista de empleados para ver a quién le corresponde ese número. De modo que podemos decir que la información se almacena en la corriente eléctrica, y que son los interruptores los que le confieren significado.

Nuestra analogía nos ha proporcionado una imagen sencilla de cómo se representa la información en un ordenador: para el ordenador sólo se trata de patrones de voltaje (es decir, las luces están ON u OFF); pero a nosotros, seres humanos, nos resulta más fácil considerar estos patrones como números binarios. Todo se reduce a una cuestión de representación. Si ahora piensa que "1010" es el código que significa "Pablo", entonces podrá empezar a comprender la relación que guarda todo con el código de lenguaje máquina. En el próximo capítulo veremos cómo se usan los números binarios para representar información en un ordenador personal.



Acelerando
Estos tres breves programas, uno para el ZX Spectrum, otro para el BBC Micro y el tercero para el Commodore 64, demuestran la diferencia en cuanto a velocidad de operación entre el BASIC y el código de lenguaje máquina para visualizar ya sea el juego de caracteres completo (Commodore y BBC) o bloques de color (Spectrum) en la pantalla

Spectrum

```
1 REM******************************
10 REM*** CODIGO L/M SPECTRUM ***
11 REM* NO LISTAR LINEA 1         *
12 REM* DESPUES DE EJECUTAR PROG*
13 REM*****************************
99 REM*****************************
150 LET PTR=23635: LET SA=PEEK
(PTR)+(256*PEEK (PTR+1))+7
200 BORDER 2
350 DATA 1,0,3,17,0,88,33,0,0,
237,176,201
400 FOR X=0 TO 11
500 READ MC
600 POKE SA+X,MC
700 NEXT X
1000 LET OFFSET=0
1100 FOR X=0 TO 1 STEP 0
1200 POKE SA+7,OFFSET
1300 LET FALSO=USR SA
1400 LET OFFSET=OFFSET+13
1500 IF OFFSET>=256 THEN LET
OFFSET=OFFSET-256*INT(OFFSET/256)
1600 NEXT X
1700 STOP
1799 REM******************************
1800 REM*GUARDAR PROG ANTES DE EJECUT*
1801 REM*GUARDAR PROG ANTES DE EJECUT*
1802 REM******************************
```

BBC Micro

```
100 REM*******************************
*****************BBC******************
**************************************
149 REM*******************************
150 REM*        CODIGO L/M BBC        *
151 REM*******************************
200 MODE 4:*TV 254
300 GOSUB 30000
700 FOR P=1920 TO 6079
800 K=K+1:IF K>2679 THEN K=1920
900 ? (HIMEM+P)=(K+47232)
1000 NEXT P
1100 PRINT TAB(13);"ESO ERA BASIC"
1200 INPUT" PULSE RETURN PARA VERSION
CODIGO LENGUAJE MAQUINA ",A$:CLS
1300 FOR L=0 TO 15:FOR B=0 TO 255 STEP L
1400 ? (SA)=LS:? (SA+1)=HS
1500 ? (FA)=LF:? (FA+1)=HF
1600 FALSO=USR(PSTRT)
1700 VDU 30
1800 NEXT B,LP
1900 END
30000 REM******S/R CARGADORA LM******
30010 K=1919:PSTRT=PAGE+8:VSTR=HIMEM+1920
30020 HS=INT(VSTR/256):LS=VSTR-256*HS:LF
=LS+56:HF=HS+2:SA=114:FA=116
30100 DATA 50,169,32,197,112,48,4,240,2,
133,112,165,112,32,227,255
30110 DATA 230,114,208,2,230,115,165,116,
197,114,208,7,165,117
30120 DATA 197,115,208,1,96,230,112,169,
128,197,112,208,224
30130 DATA 169,32,133,112,208,218,96,96,96
30150 READ ZZ
30160 FOR BY=PSTRT TO PSTRT+ZZ
30170 READ MC:? (BY)=MC
30180 NEXT BY
30200 RETURN
30299 REM*******************************
30300 REM*    ¡¡¡GUARDAR ANTES DE EJECUTAR!!! *
30301 REM*******************************
30399 REM*******************************
30400 REM*     ¡¡NO LISTAR LINEA 100 DESPUES  *
30401 REM*          DE EJECUTAR PROGRAMA!!    *
30402 REM*******************************
```

Commodore 64

```
99 REM ***********************
100 REM*CODIGO L/M COMMODORE *
101 REM***********************
200 PRINT CHR$(147) :REM LIMPIAR PANTALLA
300 PRINT "        ESTO NO LLEVARA MUCHO"
400 GOSUB 60000
500 PRINT CHR$(147);CHR$(5) :REM CLS Y BLANCO
600 CC=0
700 FOR P=SM TO FM
800 POKE P,CC:POKE P+OF,CL
900 CC=CC+1: IF CC>255 THEN CC=0
1000 NEXT P
1100 PRINT TAB(13);"ESO ERA BASIC"
1200 INPUT" PULSE RETURN PARA VERSION
CODIGO LENGUAJE MAQUINA ";A$
1300 FOR LP=1 TO 9:FOR B=0 TO 255 STEP LP
1400 POKE SA,LS:POKE SA+1,HS
1500 POKE FA,LF:POKE FA+1,HF
1600 POKE BA,B:POKE CH,0
1700 SYS AA
1800 NEXT B,LP
1900 END
60000 REM******S/R CARGADORA LM******
60010 SM=256*PEEK(648):OF=55296-SM:FM=SM
+999:BD=53280:SC=BD+1:CS=8:CB=6:CL=0
60020 POKE BD,CB:POKE SC,CS
60030 LS=0:HS=PEEK(648):LF=232:HF=HS+3:SA
=251:FA=253:BA=250:CH=2
60100 DATA 850,885,169,0,170,165,250,133,
2,165,2,129,251
60110 DATA 230,251,208,2,230,252,165,253,
197,251,208,7,165,254
60120 DATA 197,252,208,1,96,230,2,208,229,
240,223
60150 READ AA,ZZ
60160 FOR BY=AA TO ZZ
60170 READ MC:POKE BY,MC
60180 NEXT BY
60200 RETURN
60299 REM*******************************
60300 REM* GUARDARLO ANTES DE EJECUTARLO *
60301 REM*******************************
```

Bill Gates: el impulsor del estándar

En menos de una década, Microsoft se ha convertido en el proveedor de software para microordenadores más influyente del mundo



Cortesía de Microscope

Tony Sleep

La empresa Microsoft, que ahora representa un negocio multimillonario en dólares, es la clásica historia de unos entusiastas a quienes les ha ido bien. Bill Gates, que a los 28 años preside la junta directiva, en 1972 sólo era un aficionado con talento.

En la escuela superior de Seattle (Estados Unidos), donde la asociación de padres y educadores tuvo la buena idea de equipar a los estudiantes con un terminal de tiempo compartido acoplado al popular miniordenador DEC PDP-11, Bill aprendió el funcionamiento de los ordenadores. Fue a la Universidad de Harvard y, después de graduarse, se inició en los negocios en Bellevue en compañía de su condiscípulo y amigo Paul Allen. La firma que crearon se llamó *Traff-O-Data*, y su trabajo consistía en supervisar el flujo del tráfico, contratados por las autoridades públicas de Seattle. Era un momento trascendental en el desarrollo del microordenador: los primeros microprocesadores estaban haciendo su aparición y quienes tenían imaginación y entusiasmo veían un gran futuro para dispositivos como el 4004 de Intel y, posteriormente, el chip 8080. Ya entonces Bill estaba muy familiarizado con el DEC PDP-11 y uno de sus primeros trabajos fue rastrear errores en este ordenador. Pensó que sería una buena idea adaptar su BASIC para utilizarlo con el 8080. No tenía ningún sistema de desarrollo y la primera vez que acopló el código y la máquina fue cuando Bill llevó las cintas a Altair, en Albuquerque (Nuevo México). Funcionó a la primera. Y así nació el MBASIC, que desde entonces ha sido el estándar a superar.

Microsoft fue adquiriendo fama de casa de software con experiencia en dotar de sistemas operativos a nuevos ordenadores, rellenar la caja vacía, por decirlo así, y la IBM se puso en contacto con Gates para solicitar su consejo acerca de cómo especificar y equipar un ordenador personal para un único usuario. Inicialmente, Gates sugirió que Gary Kildall, de la Digital Research, el hombre encumbrado por el éxito del CP/M, era la persona adecuada para ese trabajo. Pero la IBM volvió a dirigirse a la Microsoft. Ésta reescribió el PASCAL, FORTRAN y MBASIC para su adaptación a los 16 bits y también creó el BASIC GW, con sus capacidades ampliadas para música y gráficos.

Al mismo tiempo, Gates comprendió que un sistema operativo para múltiples usuarios, poco cuidado pero poderoso, creado por los Laboratorios Bell, se podía adaptar muy bien a los micros más potentes basados en los nuevos microprocesadores de 16-32 bits, y transformó el Unix en el Xenix. Tanto Tandy como Apple adoptaron el Xenix en 1983 para sus propios modelos de 16-32 bits. Incluso Microsoft colaboró en gran medida en la creación más reciente de Apple: el Mackintosh.

Microsoft también tiene una firma que se está introduciendo en el mercado para aficionados. En 1981 se creó ASCII-Microsoft, con un japonés joven y perspicaz, Kay Nishu, para vender su sistema operativo y su BASIC a los fabricantes orientales de la nueva generación de micros, como el NEC PC 8201 y el Tandy Modelo 100. El estándar MSX común surgió a partir del deseo de los fabricantes japoneses de un modelo común, no sólo en lo que concierne a lenguajes, sino también en cuanto a interfaces para aquellos periféricos personales deseables como son plotters e impresoras en color, lápices ópticos, palancas de mando, mandos de bola, brazos-robot, sintonizadores de FM, etc. Parece ser que también tendremos muy pronto un formato de disco estándar de Microsoft que permitirá la transferencia de datos entre los tres medios operativos principales: MSX, MS-DOS y Xenix. Con su interés por un software de fácil uso, como ilustran productos como las ventanas de pantalla y el ratón, Microsoft tiene un brillante futuro.

Directrices

En 1970, a los 28 años, Shiina Takayoshi creó Sord Corporation (40 millones de dólares de ventas en 1982). Redactó 11 principios para que sirvieran de ayuda en el gobierno de su nueva empresa de informática. He aquí 3 de ellos:

"La empresa tiene una deuda, en primer lugar, con la humanidad."

"La empresa debe esforzarse al máximo por determinar qué productos y servicios son los mejores para la sociedad, y darlos a un costo razonable."

"No debe haber solución de continuidad entre el trabajo y la administración. Todas las personas de la empresa se respetarán mutuamente y cooperarán al beneficio de todos."

Estándar industrial

El BASIC (*Beginners' All-purpose Symbolic Instruction Code:* código simbólico de instrucciones para fines generales destinado al principiante) fue desarrollado en 1965 en el Darmouth College (Estados Unidos) por J. Kemeny y T. Kurtz, adelantándose, por tanto, al microprocesador en siete años al menos. Aunque se han creado muchas versiones de este lenguaje, el MBASIC, que es la de Microsoft, se ha reconocido como el estándar para la industria.

La fama de Microsoft se creó a partir del éxito del MBASIC y se consolidó con la producción de un serio competidor del CP/M de la Digital Research, el MS-DOS, un sistema operativo diseñado para aplicarlo a una amplia gama de microordenadores.

Siguiendo el camino iniciado por Xerox con su sistema terminal Star, que después Apple desarrollaría con el Lisa, en la actualidad Microsoft se ha diversificado ligeramente y ha producido un paquete que combina el software con un dispositivo de hardware necesario para su operación: ventanas MS y el ratón. El ratón de Microsoft, al igual que los otros dos de la competencia, utiliza una disposición parecida y un mando de bola acoplado con dos selectores para desplazar el cursor a través de la pantalla

Editorial Delta, S.A.



10025
9 788485 822836